UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE QUITO

CARRERA:

INGENIERÍA ELECTRÓNICA

Trabajo de titulación previo a la obtención del título de:

INGENIEROS ELECTRÓNICOS

TEMA:

SISTEMA DE CONTROL EN RED INALÁMBRICO (WNCS) MEDIANTE VISIÓN ARTIFICIAL Y HERRAMIENTA DE MENSAJERÍA TELEGRAM, PARA SISTEMAS DE NIVEL

AUTORES:

BRYAN DAVID SÁNCHEZ POZO

ALEX EDUARDO TUPIZA BASTIDAS

TUTOR:

CARLOS GERMÀN PILLAJO ANGOS

Quito, septiembre del 2020

CESIÓN DE DERECHOS DE AUTOR

Nosotros, Bryan David Sánchez Pozo y Alex Eduardo Tupiza Bastidas con documentos de identificación N° 1752329468 y N° 1726790668 respectivamente, manifestamos nuestra voluntad y cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del trabajo de titulación intitulado: SISTEMA DE CONTROL EN RED INALÁMBRICO (WNCS) MEDIANTE VISIÓN ARTIFICIAL Y HERRAMIENTA DE MENSAJERÍA TELEGRAM, PARA SISTEMAS DE NIVEL, mismo que ha sido desarrollado para optar por el título de: Ingenieros Electrónicos, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En aplicación a lo determinado en la Ley de Propiedad Intelectual, en nuestra condición de autores nos reservamos los derechos morales de la obra antes citada. En concordancia, suscribimos este documento en el momento que hacemos entrega del trabajo en formato digital de la Universidad Politécnica Salesiana.

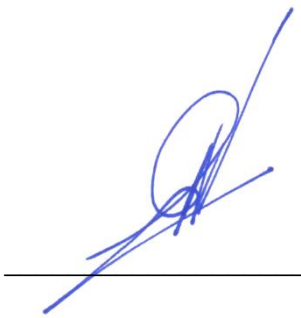
.....	.....
Bryan David Sánchez Pozo	Alex Eduardo Tupiza Bastidas
Cédula: 1725329468	Cédula: 1726790668

Quito, septiembre del 2020

DECLARATORIA DE COAUTORÍA DEL DOCENTE TUTOR

Yo declaro que bajo mi dirección y asesoría fue desarrollado el Proyecto Técnico SISTEMA DE CONTROL EN RED INALÁMBRICO (WNCS) MEDIANTE VISIÓN ARTIFICIAL Y HERRAMIENTA DE MENSAJERÍA TELEGRAM, PARA SISTEMAS DE NIVEL, realizado por Bryan David Sánchez Pozo y Alex Eduardo Tupiza Bastidas, obteniendo un producto que cumple con todos los requisitos estipulados por la Universidad Politécnica Salesiana, para ser considerados como trabajo final de titulación.

Quito, septiembre del 2020



Carlos Germán Pillajo Angos

Cédula de identidad: 1709255119

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

El presente trabajo explica el proceso realizado para implementar el control inalámbrico LQG a una planta de nivel, la planta consta de un tanque de llenado y un tanque de suministro que emula el suministro y consumo de combustible dentro del tanque de llenado usando la tarjeta Raspberry PI 3, la cual ejecuta el monitoreo de nivel por medio de visión artificial, procesa las imágenes en tiempo real para controlar de manera efectiva el proceso de la planta. Además, se desarrolló el accionamiento inalámbrico del actuador por medio del módulo ESP8266 de Arduino que comanda el servo motor acoplado a una válvula mecánica de media vuelta, la cual controla el flujo de salida del combustible del tanque de llenado.

La implementación del proyecto consta de tres fases principales, la primera fase es la generación del modelo matemático de la planta mediante IDENT de Matlab, la segunda fase es la implementación del algoritmo de control LQG a la planta y como tercera fase el desarrollo de monitoreo remoto mediante la herramienta Telegram que trabaja con el servidor de la misma aplicación, es decir puede funcionar desde cualquier dispositivo inteligente con acceso a internet y a la aplicación móvil.

Los elementos de control de la planta interactúan por medio del software de programación gráfica Node Red, el mismo que utiliza el protocolo MQTT para la comunicación de los denominados clientes con el servidor (broker) por medio del concepto de publicación y suscripción a un determinado tema.

ABSTRAC

The present work talks over the process to implement the LQG wireless control to a level plant, the plant consists of a filling tank and a supply tank that emulates the supply and consumption of fuel inside the filling tank using the Raspberry PI card. 3, which performs level monitoring using machine vision, processes the images in real time to effectively control the plant process. In addition, the wireless actuator drive was developed using the Arduino ESP8266 module that controls the servo motor coupled to a half-turn mechanical valve, which controls the flow of fuel out of the filling tank.

For the execution of the project, it was divided into three fundamental parts, the first being the obtaining of the mathematical model that describes the behavior of the plant using Matlab ident, the second is the implementation of the LQG (Linear Quadratic Gaussian) control algorithm to the plant, and the third is monitoring remote by means of the Telegram tool that works with the server of the same application, that is, it can work from any smart device with access to the internet and in turn to the mobile application.

The control elements interact through the Node Red graphical programming software, which uses the MQTT protocol for the communication of so-called clients with the server (broker) through the concept of publication and subscription to a certain topic.

INTRODUCCIÓN

En la industria uno de los procesos más importantes es el control de nivel, por lo que el tener un ineficiente control puede tener graves resultados. En el diseño de hardware se describe los elementos que conforman el montaje de la planta, esta constará de un contenedor transparente para el almacenamiento, la extracción se realizará por medio de una servo-válvula en la salida que desalojará el líquido y para el llenado del líquido se utilizará una electrobomba sumergible que succiona el líquido desde el depósito hacia el recipiente de almacenamiento.

El modelo matemático de la planta se genera mediante la adquisición de datos experimentales del proceso, utilizando el sensor ultrasónico HC-SR04, la función de transferencia se consigue con el método de caja negra manejando el software Matlab.

El desarrollo del software que se utiliza para el control inalámbrico y el monitoreo remoto dependen de una cámara web para adquirir el nivel, una tarjeta raspberry en la cual se incorporará el algoritmo de control LQG y la conexión con Telegram, en la cual existe una herramienta llamada bot, el cual será previamente programada con las acciones necesarias para brindar información textual y fotos del nivel de líquido presente en la planta, para lograr intercambiar mensajes e imágenes a tiempo real de una manera remota y fácil de usar.

El protocolo de comunicación funciona con topología tipo estrella utilizando el protocolo de comunicación MQTT (Message Queue Telemetry Transport), el protocolo tiene un servidor (bróker mosquitto) que interactúa con el software de programación gráfica Node Red, además gracias a sus librerías facilita la comunicación de los clientes en la red.

Las pruebas de control se realizan con iluminación artificial ya que la calibración del color sensado se realiza al principio del proceso, los resultados que arroja el proyecto nos permiten captar el error de medición que hay en la adquisición de nivel por visión artificial, además se puede comprar la diferencia que existe entre un control PID (Proporcional Integral Derivativo) y un control LQG aplicados a un proceso de nivel por medio de la prueba de Wilcoxon.

CAPÍTULO 1

ANTECEDENTES

El siguiente capítulo contiene el planteamiento del problema a resolver, la justificación, el objetivo general y los objetivos específicos.

1.1 Planteamiento del problema

En la industria los sistemas de nivel mayormente son una combinación de indicadores mecánicos, y análogos, en estos procesos la eficiencia de los productos que se utilizan está en función del tiempo transcurrido y las horas reales de funcionamiento, es decir no tienen la oportunidad de mejorar el beneficio de anticipar problemas antes de que ocurran, por lo tanto no están al tanto de cualquier necesidad que pueda causar daños en el entorno del proceso y en todos sus elementos.

Los sistemas de nivel que existen en la industria generalmente no realizan monitoreo de forma remota del consumo de recursos como agua o combustible, por esta razón el costo del tiempo de inactividad no planificado en el mundo industrial puede ser abrumador, entonces las empresas industriales pierden grandes cantidades de dinero por hora (IDbox RT, 2018), por lo que el control que se ejerce en diferentes sistemas industriales actualmente no tiene la conectividad virtual que permita contar con un sistema de monitoreo remoto, de hecho, existe dificultad para tomar registros de un proceso desde cualquier lugar y en el momento exacto, mediante la recopilación de datos de forma automatizada (TECNOLOGÍA PARA LA INDUSTRIA, 2020) .

En sistemas que requieren control de nivel el objetivo principal es conservar el nivel de líquido en una referencia constante, de tal manera que se puedan realizar acciones como mezclas homogéneas, prevenir desperfectos en activación de bombas, evitar derrames trabajando con flujos definidos en el proceso (BANERJE, 2015), por lo tanto para prevenir daños irreversibles en equipos y pérdidas en el proceso de producción, un tipo de control común no resulta del todo eficiente y en muchas de las ocasiones no permiten a los distintos sistemas trabajar con toda su capacidad, limitándolos de gran manera. Es entonces que se deben tomar a consideración algunos factores con el fin de realizar un control óptimo y adecuar el consumo de tiempo, facilitando el mantenimiento.

Además del control LQG es necesario contar con un monitoreo adecuado de nivel, con el fin de poder alertar al operador si existe algún problema, daño o falla en la planta. Este tipo de advertencias generalmente son de manera visual (luces indicadoras) o sonora (bocinas, sirenas) y usualmente se necesita un operador se encuentre muy cerca del proceso monitoreándolo constantemente, por lo que contar con un sistema capaz de permitir que las personas visualicen los datos desde cualquier lugar, es ideal para este tipo de situaciones, así el usuario tiene acceso a la información usando un navegador e internet aprovechando el uso de nuevas tecnologías y dispositivos móviles (OMEGA ENGINEERING, 2015) que serían de gran utilidad para el personal encargado de la planta.

Por lo anteriormente expuesto surge la pregunta si existe una alternativa de solución para los sistemas de control de nivel que utilicen monitoreo remoto y además utilice algoritmo de control óptimo.

1.2 Justificación del proyecto

En la industria uno de los procesos más importantes es el control de nivel, por lo que emplear un control ineficiente de este proceso desencadena en una afección a la producción del mismo. Por estas razones el monitoreo de nivel debe ser permanente para advertir al encargado del proceso posibles anormalidades en el flujo de líquido, en muchos procesos industriales se necesita establecer un nivel de líquido con demasiada precisión, es decir que un simple control ON-OFF no es suficiente para la acción que requiere el proceso de llenado (Jarquin & Zepeda, 2017)

El implementar un sistema de control inalámbrico mediante visión artificial aporta al costo en elementos de precisión, alta robustez dependiendo de la cámara que se emplee, gran confiabilidad, así como también equilibrio en el aspecto mecánico. Los sistemas de visión artificial se basan principalmente en cámaras de carácter industrial capaces de adquirir imágenes procesables por un controlador para analizar y calcular diferentes características del líquido en este caso con el fin de accionar los elementos que se encargan del control. (COGNEX.COM, 2020). Con el uso de visión artificial con cámara web como sensor, se pueden tener aplicaciones tan diversas enfocadas en aumentar la producción y reducir los costes, ofreciendo una alternativa a los típicos sensores ya existentes, además de que en los distintos líquidos se podrán detectar

contaminantes, materias extrañas y cualquier elemento que puede ser capaz de dañar o perjudicar al proceso.

El sistema de control inalámbrico y monitoreo remoto contará con el uso de la visión artificial con el fin de obtener el dato del nivel de líquido en la planta, y mediante la herramienta de mensajería Telegram se realizará una revisión constante del proceso, para lograr intercambiar mensajes e imágenes a tiempo real, así el usuario podrá conocer el nivel de líquido y el estado físico de la planta desde cualquier dispositivo móvil o una computadora conectados a una red, de una manera sencilla y fácil de usar.

Para desarrollar un sistema de control que pueda realizarse de manera inalámbrica, es preciso que el algoritmo de visión artificial y el controlador estén comunicados por lo que se implementará una red con topología tipo estrella con MQTT .Las redes inalámbricas son aún más desafiantes para control de retroalimentación, así como diferentes demoras (latencias) debido al tráfico congestión y colisión en la red (CHEN, McKERNAN, IRWIN, & SCALON, 2008). La red el retraso degrada el rendimiento y puede provocar inestabilidad (MATIAKIS, HIRCHE, & BUSS, 2009), por lo que el diseño del controlador es de suma importancia.

Por lo tanto el tipo de control a implementar debe ser capaz de controlar la variable de una manera eficiente y precisa, puesto que un control eficiente en la industria incluye constantes mejoras en el porcentaje de eficacia, reducción en el uso de energía no renovable, reducción de materia prima desechable y control permanente sobre los niveles de seguridad. (INFAIMON, 2017).

1.3 Propuesta de solución

Con el fin de poder realizar el control y monitoreo de nivel de manera inalámbrica se tiene pensado el uso de la aplicación Telegram Messenger, el empleo de visión artificial (Sistema de monitoreo), y el desarrollo del controlador LQG.

El controlador estará alojado en la raspberry Pi que funcionará como el servidor de la planta, será desarrollado con el software NodeRed, para el sistema de monitoreo se realizará un algoritmo de visión artificial desarrollado en Phyton, de tal manera que sea posible identificar el nivel de líquido que se encuentre en el tanque de la planta,

este dato podrá ser enviado a un usuario cuando este lo requiera mediante la aplicación de Telegram.

Para la comunicación inalámbrica entre la raspberry, el módulo ESP8266 y Telegram se utilizará el protocolo de comunicación MQTT. Donde en un extremo de la red se encontrará el sensado del proceso con el algoritmo de visión artificial desarrollado en la raspberry, la cual tendrá instalado el bróker (servidor) quien se encarga de enviar datos a clientes en la red a través de suscripciones, es decir obtendrá el dato de nivel del líquido, con el fin de enviar toda esta información hacia el controlador.

El monitoreo mediante Telegram será totalmente aislado del controlador, por lo que el broker enviará la información hacia los dispositivos que cuenten con la aplicación de Telegram, si el usuario así lo requiere, esto es posible ya que la API cuenta con su propio servidor.

1.4 Objetivos

1.4.1 Objetivo general

Desarrollar un sistema de control en red inalámbrico, de nivel utilizando visión artificial para monitorearlo remotamente desde cualquier dispositivo mediante Telegram.

1.4.2 Objetivos específicos

- Realizar el sistema de control de nivel cuyo modelo matemático sea obtenido mediante herramientas de Matlab para el desarrollo de un algoritmo de control eficiente.

- Estructurar un sistema de comunicación inalámbrico mediante el protocolo MQTT para que relacione el sensado de nivel por visión artificial.

- Gestionar los datos obtenidos del nivel de la planta a través de la aplicación Telegram para su monitoreo remoto.

- Realizar pruebas de control inalámbrico aplicando algoritmo de control LQG para que la latencia del proceso sea mínima.

CAPÍTULO 2

MARCO TEÓRICO

En este capítulo se habla sobre los tipos de control en una planta de nivel, las funcionalidades del monitoreo remoto por Telegram en un sistema industrial. También se describe las propiedades de visión artificial al sensar el combustible y del software de programación gráfico Node Red afondando el estudio en la facilidad que tiene al conectar tecnologías de diferente tipo y las ventajas de la fusión con el protocolo MQTT, así como del monitoreo a nivel local usando su herramienta dashboard en el presente proyecto.

2.1 Tipos de control para sistemas de nivel

El control de nivel de líquidos en el presente proyecto emula un sistema de segundo orden ya que se utiliza 2 tanques para que la estructura física de la planta sea funcional, el controlador de la planta es de tipo analógico ya sea PID o LQG y tiene la función de aplicar su algoritmo sobre la señal de error. El resultado del controlador es enviado al actuador dentro de la planta, este actuador maniobra mecánicamente el caudal que pasa del tanque de llenado al tanque de suministro, con el objeto de mantener el nivel en la señal de referencia, en resumen la planta cuenta con 2 maneras eficientes de controlar y experimentar el nivel de combustible en la planta.

2.1.1 Wireless Network Control Systems

Las redes inalámbricas actualmente son usadas directamente en el control de sistemas industriales, existen ventajas y desventajas en el uso de protocolos inalámbricos para estas aplicaciones, pero la situación mejora gracias a procesos que relacionan conocimientos en control, comunicación e informática.

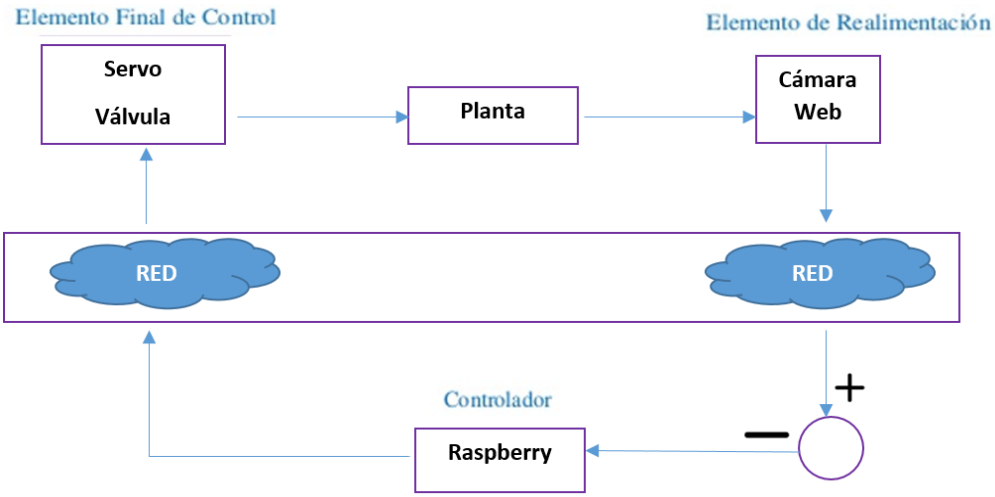
El control con retroalimentación inalámbrica en tiempo real demanda que la comunicación entre controladores, sensores y actuadores en una planta deba ser compartida con otras aplicaciones. La compartición permanente de datos a través de internet tiene correlación con la seguridad, estabilidad y rendimiento. Las características de comunicación inalámbrica tratan de mejorar los recursos de

comunicación que son primordiales en los sistemas que utilizan un control en lazo cerrado en tiempo real.

Las falencias que existen en las redes inalámbricas, ponen en duda la implementación de los controladores con realimentación, es decir, los WNCS todavía están en progreso y más aún las implementaciones en el campo industrial. (Pillajo & Hincapie, 2018)

Todos los elementos de la planta tienen una disposición lógica para realizar el control de la misma, en la Figura 2.1 se puede apreciar el diagrama de bloques del control en lazo cerrado del proceso de la planta con todos sus elementos relacionados.

Figura 2.1. Diagrama de bloques del control WNCS en lazo cerrado de la planta



Elementos del control en lazo cerrado, (Sánchez Bryan & Tupiza Alex)

2.1.2 Control PID

El controlador PID funciona tomando en cuenta la diferencia en la señal de retroalimentación, esto quiere decir que se controla que el estado actual de nivel no se diferencie mucho con el nivel deseado de referencia. Por lo contrario si el error es mínimo, quiere decir que el controlador PID es óptimo.

El controlador proporcional funciona de una forma directamente proporcional a la señal de error y trata de acortar el error general del sistema.

La acción derivativa está proporcionada a la derivada de la señal del error adquirido, es otra forma de precisar la velocidad del error, es decir si el proceso se mueve a una alta velocidad al set point, el sistema está sujeto a la propiedad de la inercia por lo que obtendrá un sobre impulso y oscilaciones respecto a la referencia. El controlador evita este problema registrando la velocidad con la que el nivel se aproxima a la referencia para tomar acción sobre el elemento que sea necesario para que el nivel se acerque a la referencia evitando oscilaciones. (Gonzalo, 2016)

2.1.3 Control LQG

El control LQG (Linear Quadratic Gaussian) es un tipo de controlador óptimo obtenido como la combinación de un estimador lineal cuadrático (LQE) con un regulador lineal cuadrático. (LQR)

Tiene como característica su gran inmunidad al ruido blanco y se aplica a los sistemas variables en el tiempo lineales, sistemas no lineales y a los sistemas lineales invariantes en el tiempo. (Athans, 1971)

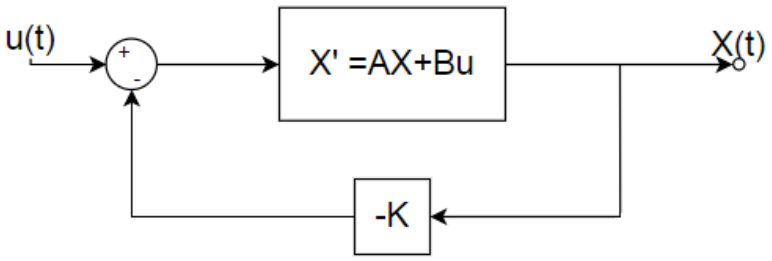
Tanto el estimador, como el regulador pueden ser diseñados y se calculan de forma independiente.

2.1.4 Control LQR

El controlador LQR ofrece un regulador que se controla de forma óptima a partir de la minimización de la función de coste, con el fin de mejorar y optimizar la eficiencia de un proceso. Es un tipo de control óptimo que depende de la habilidad del diseñador para ajustar lo más posible las necesidades de control de cualquier sistema. (Menendez & Simon, 2004)

El diagrama de bloques del Controlador LQR se muestra en la Figura 2.2.

Figura 2.2. Diagrama controlador LQR



Funcionamiento del control LQR, (Sánchez Bryan & Tupiza Alex)

2.1.5 Filtro de Kalman

El filtro de Kalman es denominado como un algoritmo que se dedica a la estimación el estado actual de cualquier sistema a partir de datos medibles. Se comporta como un algoritmo predictivo que se encarga de estimar los estados y de suministrar al controlador lineal cuadrático una señal menos afectada por perturbaciones o ruido.

Para la realización del filtro de Kalman se requieren principalmente dos pasos: el primero anuncia el estado del sistema, y el segundo utiliza las medidas de ruido para configurar la estimación del estado en el proceso. (MathWorks, 2020)

Para el diseño del controlador LQG, el modelo del proceso se representa mediante ecuaciones de entrada como se muestra en la ecuación 2.1 y salida como se muestra en la ecuación 2.2

$$\dot{x} = A\,(t) + (t) + (t) \qquad \text{Ec. (2.1)}$$

$$y = C\,(t) + D\,(t) + v\,(t) \qquad \text{Ec. (2.2)}$$

Donde:

A, B, C, D = Representan las matrices de estado del proceso
Wk = Ruido del sistema
Vk = Ruido del proceso

El diagrama de bloques del controlador LQG se muestra en la Figura 2.3

Figura 2.3. Diagrama de bloques del controlador LQG

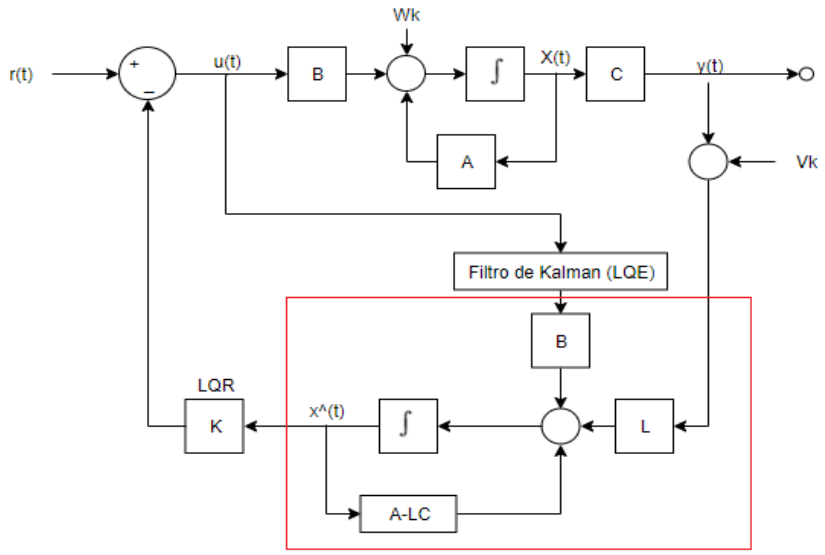


Diagrama lógico de funcionamiento de control LQG en Simulink, (Sánchez Bryan & Tupiza Alex)

Donde

L = Estimador o ganancia del filtro de Kalman (LQE)

K = Ganancia de realimentación proveniente del regulador LQR

2.2 Monitoreo remoto con la aplicación móvil Telegram

Telegram es una aplicación libre que brinda servicio de mensajería y actualmente es la única plataforma que brinda la capacidad de desarrollar e interactuar con bots, que son programas informáticos capaces de llevar a cabo tareas concretas e imitar el comportamiento humano.

El desarrollo de automatización de respuestas simples con dispositivos en línea es una de las opciones más simples y mejor pensadas cuando se trata de soluciones de problemas recurrentes, los bots poseen las características ya antes mencionadas y son implantadas en forma de monitoreo remoto en el presente proyecto. (EL ECONOMISTA, 2020)

Un bot necesita de un botFather para tener accesibilidad con todos los usuarios que accedan a la aplicación sin coste alguno para el desarrollador o el usuario accediendo a servicios de información, los mismos que se pueden utilizar para realizar el respectivo control de la planta, esta situación es viable gracias a sistemas embebidos

como la Raspberry Pi 3, basado en la arquitectura ARM, la misma que funciona como cluster de procesamiento de datos. (Baidez, 2018)

2.2.1 Bots de Telegram

Los bots de Telegram son una serie de aplicaciones de terceros que se ejecutan dentro de la aplicación de mensajería, esta herramienta se integra como si fuera una persona real con la que se puede interactuar. Los bots funcionan desde junio del 2015 , al ser bots conversacionales, por lo general no requieren una pulsación, aunque muchos también incluyen botones en sus respuestas con los que enviar comandos o peticiones sin escribirlas. (Telegram messenger, 2020)

2.3 Monitorización en Sistemas Industriales

La monitorización se encarga de la corroboración del estado de un sistema completo, el monitoreo permanente de una planta permite verificar si las máquinas necesitan mantenimiento.

IoT es una expresión que expresa el Internet Industrial De las Cosas que puede funcionar a través de funciones que son capaces de manejar datos, permitiendo una interacción inteligente de dispositivos para optimizar la producción en los procesos. (Consulting Informàtico, 2019)

El monitoreo remoto se puede aplicar en la elaboración, la explotación de minas, la fabricación de textiles, el procesamiento de alimentos; De ahí existe la necesidad de implementar novedosamente el monitoreo remoto en la optimización, mantenimiento de los dispositivos y vías de producción, restando los tiempos de inacción, este concepto viene tomando la forma de trabajo de la industria 4.0. (proximassystems.net, 2018)

2.4 Visión Artificial en Python

La visión artificial en la industria integra sensores y dispositivos de entrada y salida para controlar sistemas, es decir se puede obtener datos de cualquier proceso para desarrollarlos y programar acciones que favorezcan a la identificación del objeto principal a sensar.

La visión artificial no tiene una normativa para la regularización de su uso en nuestro país en la actualidad, pero existe la ley de Protección de Datos de Carácter Personal (Ley Orgánica 15/1999, de 13 de diciembre). La aplicación de visión artificial solo tiene fines de investigación y todas las pruebas se realizarán dentro de un laboratorio. (Justo de Frìas, 2019)

El proceso de visión artificial es considerado como una rama de inteligencia artificial, entonces el propósito es lograr que un computador imagine una escena y las peculiaridades puntuales de cierta imagen por varios métodos como procesamiento de imágenes, teoría de adquisición de colores, reconocimiento de patrones, etc. (Càceres, 2011)

2.5 Node Red

Es un software que trabaja con flujos y tiene enfoque IoT, admite definir de manera gráfica nodos de servicios, a través de protocolos estandarizados, es una herramienta liviana y visual la cual está programada en NodeJS y que puede correr en dispositivos con bajo procesamiento.

Node-Red es una nuevo instrumento de fuente abierta establecida por IBM y permite interconectar varios conceptos del Internet de las Cosas (www.techedgegroup.com, 2020), Node-Red simplifica la conceptualización de eventos en la vida real, agregar cierto valor de comprensión en nodos e integra eventos con toda clase de sistemas de mensajería como el protocolo MQTT, ya que está basado en node.js.(Ricardo Vega, 2015)

2.6 Protocolo MQTT

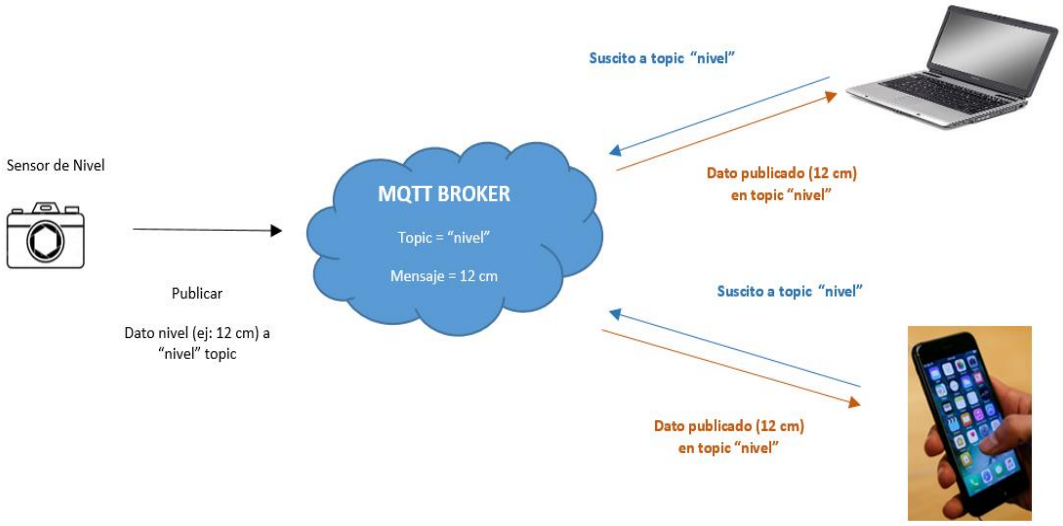
MQTT es un protocolo de transporte de datos de tipo publicador/subscriptor y está diseñado con una fácil implantación por lo que se usa en entornos, como la comunicación de máquina a máquina (M2M) e Internet de las cosas (IoT) donde la característica de ancho de banda en la red es de suma importancia.

Este protocolo sobresale en el modo de transferir paquetes a través del cable al igual que en el protocolo HTTP, además que tiene mucha facilidad para implementarse en

el caso de ser un cliente. La facilidad de uso fue una preocupación clave en el desarrollo de MQTT y lo hace perfecto para dispositivos con recursos limitados en la actualidad. (hivemq.com, 2020)

En la Figura 2.4 se puede observar la interacción de elementos en el protocolo MQTT

Figura 2.4. Protocolo MQTT



Funcionamiento lógico entre dispositivos con protocolo mqtt, (Sánchez Bryan & Tupiza Alex)

2.6.1 Broker Mosquitto

Mosquitto es un agente de mensajería instantánea dispuesto como servidor en el protocolo MQTT, es liviano y se puede usar en cualquier dispositivo inteligente. MQTT proporciona un método sencillo para interactuar en una red con topología tipo estrella, utilizando un modelo de publicación / suscripción.

El bróker mosquitto es ideal para manejar la mensajería de Internet de las cosas, facilita una biblioteca C para sumar nuevos clientes MQTT a la red, y los clientes MQTT de línea con los comandos mosquitto_pub y mosquitto_sub (Eclipse Mosquitto ™, 2020).

CAPÍTULO 3

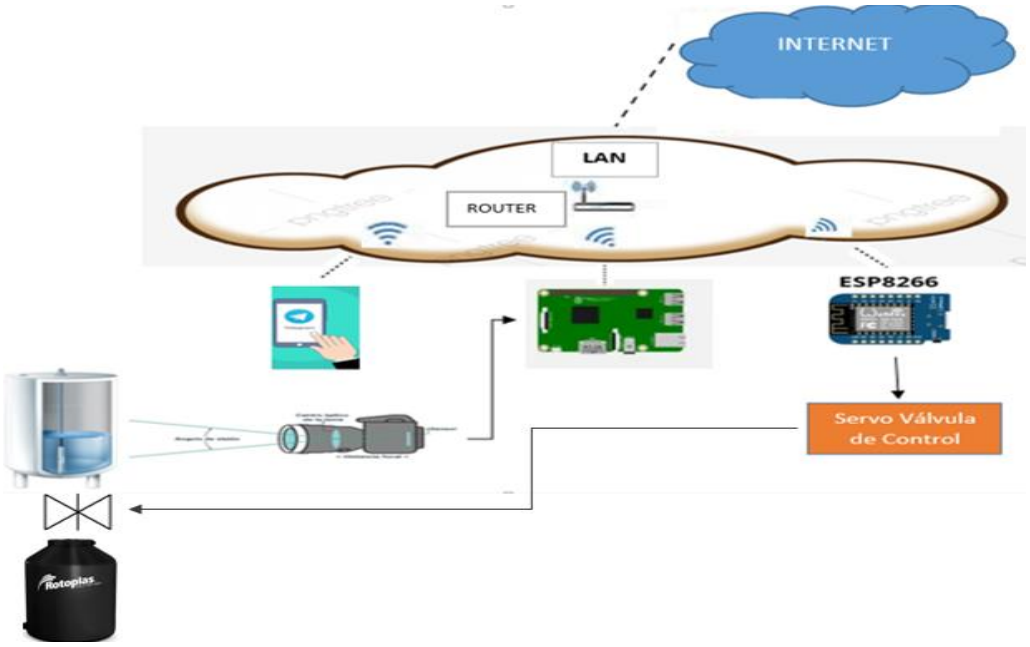
DISEÑO E IMPLEMENTACIÓN

En este capítulo se habla sobre el desarrollo del montaje y las características principales de todos elementos y sistemas que conforman la planta, así como su función dentro del proceso. También se tratará el montaje y disposición física de la estructura con las conexiones eléctricas necesarias para realizar la acción del sensado por visión artificial y el accionamiento a la servo-válvula a través del módulo ESP8266.

El software que utiliza el proyecto, principalmente emplea Matlab, Arduino, Node Red y el lenguaje de programación Python. Los mismos que se llevaron a cabo de forma progresiva; después del montaje de la estructura de la planta a escala.

El proyecto esta implementado en una planta a escala, la cual consta de un reservorio de combustible en la parte inferior y en la parte superior el tanque de llenado en el que se efectúa el control de nivel; el reservorio inferior tiene como función subministrar el combustible al tanque de llenado utilizando una bomba sumergible eléctrica con un flujo constante. En la Figura 3.1 se indica el diagrama de bloques del sistema lógico.

Figura 3.1. Esquemático del sistema lógico del proyecto



Esquema de comunicación del proyecto, (Sánchez Bryan & Tupiza Alex)

En la Figura 3.2. Se encuentra el esquemático de la conexión del sistema físico de la planta.

Figura 3.2 Diagrama de bloques del sistema físico del proyecto

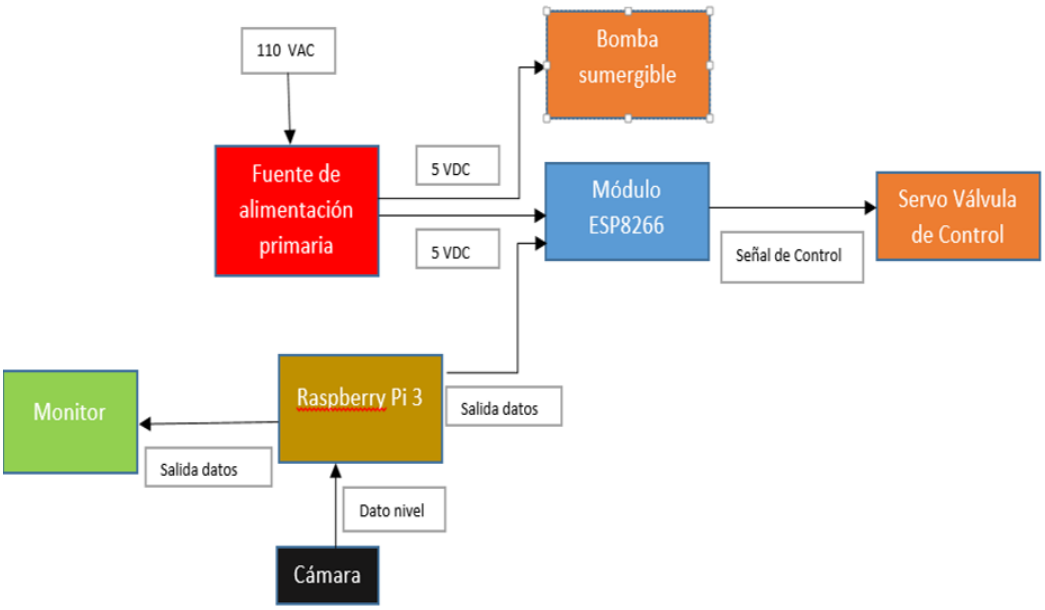


Diagrama lógico de funcionamiento de los elementos físicos del proyecto, (Sánchez Bryan & Tupiza Alex)

3.1 Componentes y Sistemas

Para la implementación y trabajo de la planta a escala con control de nivel inalámbrico se requiere los componentes y sistemas que se detallan a continuación, cada uno de ellos cumplen una función específica en el proceso.

3.1.1 Recipientes de alojamiento para combustible

En un proceso industrial normalmente el flujo de combustible va del tanque de almacenamiento hacia donde se necesite suministrar la materia prima, por lo que el reservorio inferior en la planta cumple la función de emular el consumo de la misma. El control de nivel debe estar sujeto a las características físicas de la planta y es esencial que el volumen que aloja el reservorio inferior pueda subministrar completamente de líquido al tanque de llenado en la parte superior así se cumple que el set point deseado pueda fijarse tanto en niveles mínimos como en niveles máximos del tanque de llenado, así como también que la bomba eléctrica no funcione en vacío por lo que podría averiarse.

En el control de nivel el set point se configura asignando el nivel en centímetros de largo, los mismos que están representados por una regleta en el tanque de llenado.

El combustible alojado en el reservorio inferior debe ser suficiente para suministrar todo el tanque de llenado por lo que se comprueba que el volumen sea el mismo en los dos recipientes, mediante el uso de la ecuación 3.1 que representa al Volumen de tanque de llenado, y también al volumen de reservorio se obtiene el siguiente resultado:

$$\text{Volumen de tanque de llenado} = \pi * r^2 * h \qquad \text{Ec (3.1)}$$

$$\text{Volumen de tanque de llenado} = \pi * (3.8\ \text{cm})^2 * 21\text{cm}$$

$$\text{Volumen de tanque de llenado} = 952.65\ \text{cm}^3$$

$$\text{Volumen de reservorio inferior de la planta} = \pi * r^2 * h \qquad \text{Ec (3.1)}$$

$$\text{Volumen de reservorio inferior de la planta} = \pi * (4.9\ \text{cm})^2 * 12.5\text{cm}$$

$$\text{Volumen de reservorio inferior de la planta} = 980.58\ \text{cm}^3$$

3.1.2 Válvula de bola ¼ de pulgada

La acción mecánica del paso de agua desde el tanque de almacenamiento hasta el reservorio en la parte inferior es realizada por la válvula de bola de diámetro de ¼ de pulgada, originalmente es utilizada para puntos hidráulicos en instalaciones de nivel industrial ya que es resistente a la corrosión, así como también al efecto de sustancias que son fuertes químicamente, como el combustible que utiliza la planta a escala.

La válvula de paso también funciona con el sistema de apertura de bola de ¼ de vuelta por lo que tiene una pérdida nula, es decir permite que el servo controle la apertura y cierre con los grados de activación del servo. Originalmente la válvula de bola consta de una manija metálica para su accionamiento mecánico, la cual será desmontada de la estructura original, dejando al descubierto únicamente el eje que acciona el paso de flujo.

3.1.3 Unidad principal de control

Para la función de control principal se utiliza la tarjeta Raspberry Pi 3 modelo B, que se encarga de la comunicación y del control del proceso; en esta placa multifuncional se ejecuta visión artificial, el protocolo de comunicación y Node red que es la interfaz gráfica de programación, las características principales de la tarjeta se muestran en la Tabla 3.1, estas son indispensables para la vialidad del proyecto

Tabla 3.1. Características técnicas de la Raspberry Pi 3 modelo B+

CPU + GPU	BCM2837B0, Cortex-A53 (ARMv8) 64-bit
RAM	1 GB LPDDR2 SDRAM
Wi-Fi	2.4 GHz y 5 GHz IEEE 802.11
Ethernet	Gigabit Ethernet 300 Mbps
GPIO	40 pines
HDMI	Si
Puertos USB	4 puertos CSI para cámara

Tabla de las características de la tarjeta Raspberry, (xataka.com, 2020)

3.1.4 Fuente de alimentación primaria

La fuente LPF2 a 250W que se encarga de suministrar energía a la bomba eléctrica, el módulo ESP8266 NODE y al motor servo HS-311 que conforman el proceso de la planta. En la Tabla 3.2 se muestra las características principales de la fuente de alimentación LPF2.

Tabla 3.2. Características técnicas de la fuente de alimentación primaria

Voltaje de entrada	100-240 [Vac]
Voltaje de salida	5 [Vdc]
Corriente de entrada	10 [A]
Corriente de salida	0.8[A]
Frecuencia	50 a 60 Hz
Potencia máxima	135[W]
Dimensiones	15cm * 10 cm

Tabla de las características de la fuente de alimentación primaria, (Sánchez Bryan & Tupiza Alex)

3.1.5 Módulo ESP8266 Node

El módulo ESP8266 NODE se ilustra en la Figura 3.3 y se encarga de transmitir los datos de control que genera el software NODE RED instalado en la placa Raspberry Pi 3, estos datos son necesarios para controlar la acción de la servo válvula para el paso

de combustible; además este módulo se encarga de alimentar al servo HS-311, así tendrá la suficiente potencia para ser accionada mecánicamente.

Figura 3.3. Módulo ESP8266 NodeMCU v2

Descripción física del módulo ESP826, (panamahitek, 2020)

En la Tabla 3.3 se muestra las características principales del módulo ESP8266 NodeMCU v2.

Tabla 3.3. Características técnicas del módulo ESP8266 NodeMCU v2

Voltaje de alimentación	5 [Vdc]
Voltaje lógico de entrada/salida	3.3 [Vdc]
Frecuencia de Reloj	80 MHz-160 MHz
Pines Digitales GPIO	17 (4 pueden configurarse como PWM)
Pin Analógico	1
Potencia de salida	+19.5 dBm
Corriente de fuga menor	10 uA

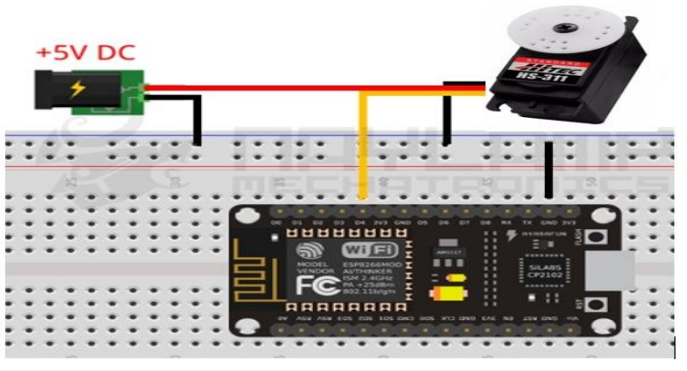
Tabla de las características de módulo ESP8266, (panamahitek, 2020)

3.1.6 Servo motor HS-311

El servo motor se integra en el acoplamiento de la válvula de bola de ¼ y pertenece al modelo HS-311, el cual fue predeterminadamente escogido por el torque o par de parada de 3.0 kg/cm que proporciona como se puede apreciar en la Tabla 3.4, la cual especifica las características del servo; para que el torque sea transmitido al eje de la válvula de paso se diseñó un acople en el software Inventor de Autodesk y se imprimió en 3D, el mismo que coincide exactamente en el medio de los 2 ejes para que no exista fuga de combustible y el par de parada no se vea afectado en el accionamiento del actuador.

Para que la alimentación del servo sea eficiente, se destaca que los servos de mediana potencia, como el HS-311 no se puede alimentar directamente de la Raspberry PI 3 y necesita una fuente de alimentación externa como el modelo cableado se muestra en la Figura 3.4.

Figura 3.4. Esquema cableado para el servo HS-311



Esquema de conexión del módulo ESP8266 con un servo, (servocity, 2020)

Tabla 3.4 Características técnicas del servo motor HS-311

Dimensiones	1.57”*0.78”*1.43” plg
Peso del producto	1.52 oz (43g)
Estilo de eje de salida	Estrías de 24 dientes
Voltaje de alimentación	4.8 - 6.0 (V)
Rango máximo de PWM	575-2460 useg
Amplitud de pulso	3-5V

Tabla de las características de motor servo HS-311, (naylampmechatronics, 2020)

3.2 Estructura de la planta a escala

La estructura emula al proceso de control, suministro y consumo en un sistema de nivel por lo que las propiedades físicas cumplen parámetros como que el reservorio de combustible debe estar en la parte inferior del tanque de llenado, ya que con la acción de la gravedad el combustible baja cuando el actuador permita el flujo o lo corte del combustible; por esta razón se utiliza 2 superficies de dimensiones 30 cm * 30 cm horizontales de madera que se disponen en forma paralela estas están sujetas con 4 cilindros de 22 centímetros de alto estilo columna para que la estructura tenga estabilidad.

El diseño de la estructura se puede apreciar en el modelado de la planta en el software Inventor en la Figura 3.5

Figura 3.5. Modelado de la estructura de la planta en el software inventor

Diseño de estructura de la planta en Inventor, (Sánchez Bryan & Tupiza Alex)

La manguera que sube desde el reservorio se apoya sobre una quinta columna rectangular de un 45 cm de alto, la misma que se encarga de dar soporte para que la planta cumpla con un único comportamiento al realizar el llenado.

3.3 Sistema Hidráulico

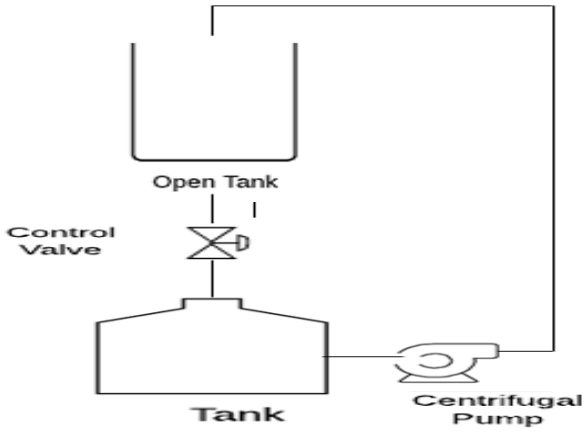
El sistema hidráulico trabaja con la presión de fluidos así que se encarga de enviar el combustible de color naranja desde el reservorio en la parte interior de la planta a escala al tanque de llenado mediante la activación de la bomba eléctrica por medio de una manguera de diámetro de 1/8 de pulgada, una vez que empiece el control el servo acoplado a la válvula de bola de ¼ de pulgada. En la Figura 3.6 se muestra los componentes del sistema hidráulico como son: servo válvula de control, mangueras, bomba sumergible eléctrica y acople “bushing” de ¼ a 1/8.

Figura 3.6. Componentes del sistema hidráulico

Fotografía de bomba eléctrica y válvula de paso, (Sánchez Bryan & Tupiza Alex)

El sistema hidráulico está representado por los símbolos de cada uno de sus elementos y esquematizado para su mayor compresión en la Figura 3.7

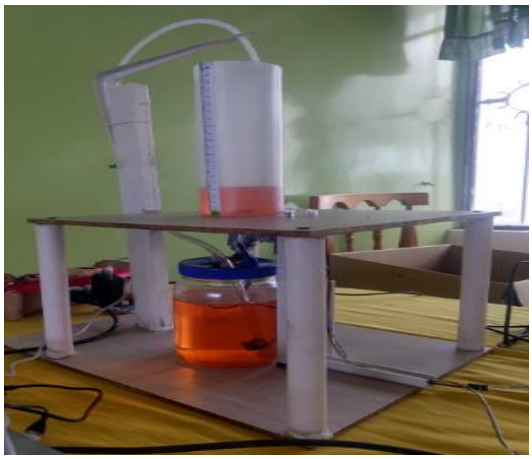
Figura 3.7. Esquema espacial de sistema hidráulico



Representación gráfica del sistema hidráulico, (Sánchez Bryan & Tupiza Alex)

La manguera que conecta el flujo en todo el sistema es de ¼ “, por lo que tiene el suficiente diámetro para que la bomba sumergible actúe con toda su fuerza, estos elementos están acoplados con una abrazadera para evitar que la bomba pierda potencia. El sistema completamente montado se puede apreciar en la Figura 3.8

Figura 3.8 Montaje del sistema hidráulico



Fotografía del montaje de la planta, (Sánchez Bryan & Tupiza Alex)

3.4 Sistema de visión artificial

La visión artificial funciona a través una cámara web, cuya función es captar y procesar la información acerca de los distintos volúmenes que arroja el tanque de llenado. La adquisición de imágenes usa una cámara web USB estándar, siempre teniendo en cuenta que la calidad y la capacidad de configuración del módulo de la cámara en la Raspberry Pi 3 sean superiores a la de una cámara web USB estándar.

La ventaja de utilizar la cámara web estándar del proyecto es que no necesita drivers propios de su software, además tiene una resolución de 800 * 600 pixeles, este dispositivo periférico se puede instalar directamente desde el repositorio de Raspbian.

La distancia a la que se coloca la cámara para lograr la adquisición del nivel debe cumplir con el límite mínimo para que la imagen procesada tenga la suficiente longitud focal, es decir el sensado con visión artificial es óptimo ya que la cámara web se coloca a una distancia de 55 cm del tanque de llenado, esta distancia dependerá del tipo de cámara con la que se implemente el proyecto así como las magnitudes de la planta a controlar, estos valores se pueden observar en la Tabla 3.5

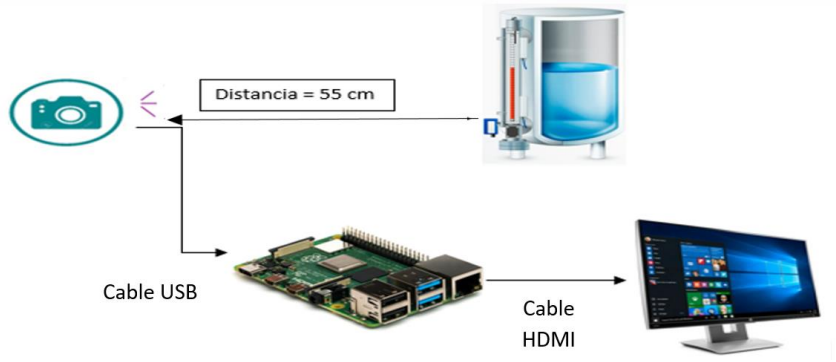
Tabla 3.5. Distancias del campo de visión según la resolución de una cámara.

Resolución horizontal	Longitud focal	Distancia máxima	Distancia mínima
2592 pixeles	2.8-8 mm	9 m	5.2 m
1280 pixeles	3.3-12 mm	6 m	2.6 m
1920 pixeles	5.1-5.1 mm	41 m	3.8 m
736 pixeles	3.3-119 mm	50 m	1.5 m
1280 pixeles	4.4-132 mm	67 m	2.6 m

Tabla de características de cámara según su resolución, (ni.com, 2020)

En la Figura 3.9 se aprecia el diagrama esquemático de la disposición de espacio de los elementos involucrados.

Figura 3.9 Diagrama esquemático de visión artificial



Disposición de elementos del sistema de visión artificial, (Sánchez Bryan & Tupiza Alex)

3.5 Software a utilizar

3.5.1 Raspbian

Raspbian se considera el sistema operativo de Debian, es gratuito y está destinado para funcionar con el hardware de la tarjeta Raspberry Pi, es una serie de algoritmos básicos y utilitarios que vienen pre compilados (debian.org, 2020). El sistema soporta cálculos con coma flotante conjuntamente con el hardware además que su implementación fue necesaria ya que no existía versión para la CPU ARMv6 que contiene el Raspberry Pi, en la Tabla 2.6 se describen los requisitos de hardware mínimos para la instalación del mismo.

Tabla 3.6. Requisitos mínimos de hardware para instalar Raspbian

Tipo de instalación	RAM(mínimo)	RAM(recomendado)	Disco duro
Sin escritorio	128 Megabytes	512 Megabytes	2 Gigabytes
Con escritorio	256 Megabytes	1 Gigabyte	10 Gigabytes

Tabla de requerimientos mínimos para instalar sistema operativo Raspbian, (debian.org, 2020)

3.5.2 Python

La Raspberry Pi 3 que se utiliza como controlador tiene el sistema operativo Raspbian, el mismo que posee a el programa Python por defecto. Se utilizó el IDE Thonny ya que es más sencillo, tiene solo las funciones básicas y esta designado para programadores principiantes en el lenguaje, este entorno viene con Python 3.7 integrado, por lo que solo se necesita un instalador simple, también puede usar una instalación de Python por separado si es necesario.

3.5.3 OpenCV

Es una serie de librerías compiladas en varios lenguajes de programación, que se desarrollaron con el objetivo de implementar visión artificial en los computadores. Implementa nuevos algoritmos para la definición concreta de imágenes que pueden ser optimizadas. (Mullo López, Molina, & Andrés, 2016)

Es una librería de software abierto y machine learning, la cual consta de una infraestructura para implementaciones de visión artificial como la detección de intrusos en vídeos o monitorización de equipamientos, estas aplicaciones se tomarán como base para el proceso de visión artificial en la planta a escala. (unpocodejava.com, 2020)

3.5.4 Arduino

Arduino es una plataforma de código abierto, la cual es flexible y fácil de utilizar para los desarrolladores, es decir permite utilizar elementos libres, vinculándose a dispositivos e interactuando a través del IDE que es propio de Arduino (xataka.com, 2020). Nos sirve para convertir la información en una acción como puede ser en esta ocasión manejar el actuador de la planta.

3.6 Identificación de modelo matemático

Los diferentes procesos de nivel que se encuentran en la industria, tienen un comportamiento debido a la disposición física de la misma y a los elementos que conforman la planta como actuadores, sensores, controladores, estos últimos describen la dinámica de la planta ya que están instalados en la misma.

La dinámica de la planta se identifica con el método de la caja negra el cual describe el funcionamiento de un sistema únicamente con entrada (flujo de combustible que entra al tanque de llenado) y salida (flujo de combustible que sale del tanque de llenado hacia el reservorio inferior), para identificar la reacción entre el estímulo y la respuesta.

Los criterios de diseño se basan en la reacción del sistema a las permutaciones en las condiciones iniciales, es decir la señal de prueba que en este caso fue accionar el servo motor cerrándolo con un ángulo diferente al inicial, la misma que actúa como una función escalón afectando la respuesta transitoria del sistema.

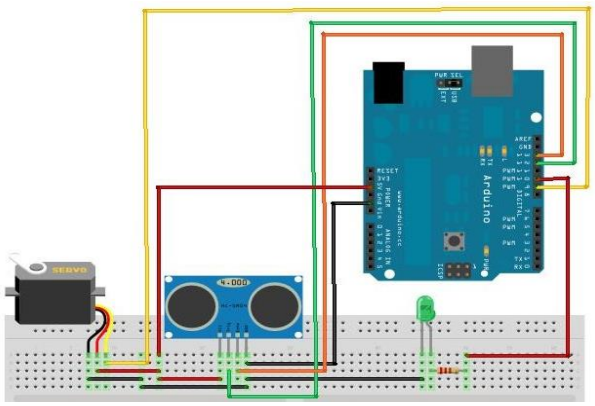
3.6.1 Adquisición de datos experimentales

El uso de la señal de prueba o función escalón existe siempre que la reacción de un proceso para una señal paso de prueba común cambie en relación al tiempo, por lo tanto el nivel en el tanque de llenado cambiará hasta que pase de un estado transitorio a un estado estable, haciendo que el nivel sea constante de nuevo.

Para la adquisición de la respuesta de la planta a través del tiempo se utilizó el sensor ultrasónico HC-SR04 conectado al Arduino uno, midiendo la respuesta de la planta a la función escalón es decir el cambio de nivel de combustible, cuando la servo válvula cambie los grados de accionamiento de 0 grados (llave abierta) a 55 grados (llave semi cerrada).

En la Figura 3.10 se encuentra la conexión del servomotor y el sensor ultrasónico para la respectiva adquisición de datos.

Figura 3.10 Conexión para adquisición de datos experimentales

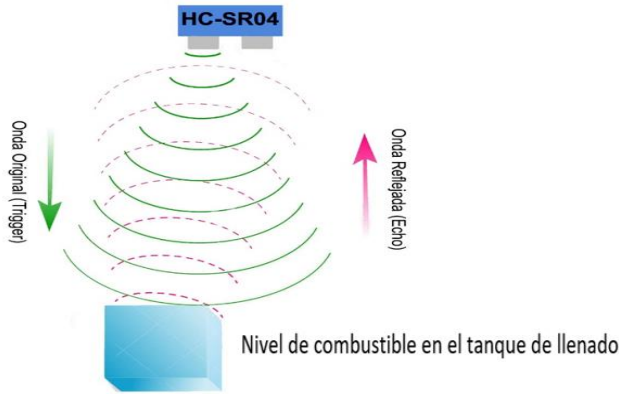


Conexión de Arduino, servo y sensor HC-SR04, (tinkercad, 2020)

El sensor ultrasónico de Arduino HC-SR04 funciona dando pulsos digitales con el disparador de pulsos (trigger) durante unos microsegundos y debemos escuchar la señal de eco (echo), una vez que reaccione al combustible que está subiendo o bajando en el tanque de llenado como se ilustra en la Figura 3.11.

Ya que el tiempo que tarda en volver el eco en µs (microsegundos) y la velocidad del sonido es de $343\,\frac{m}{s}$, se calculó cuanto tiempo se necesita para que la señal recorra un metro con la ecuación 3.2.

Figura 3.11 Funcionamiento del sensor HC-SR04 implementado en la planta



Función del sensor ultrasónico, (Sánchez Bryan & Tupiza Alex)

$$\text{Tiempo de recorrido de onda reflejada} = \frac{d}{v} \qquad \text{Ec (3.2)}$$

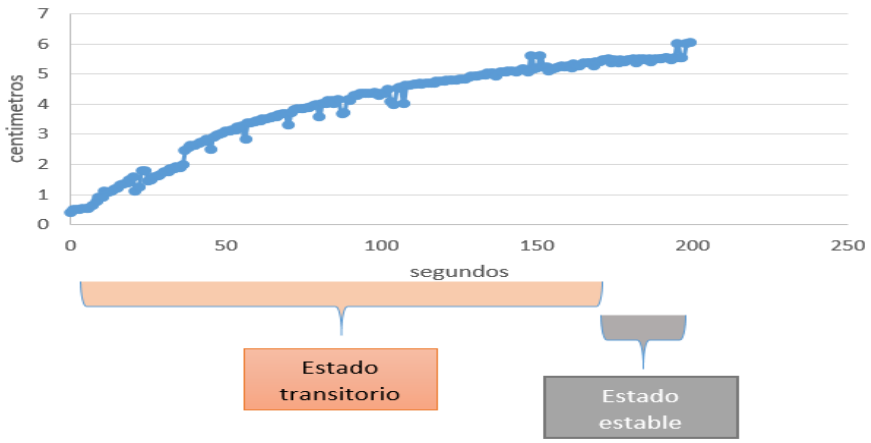
$$\text{Tiempo de recorrido de onda reflejada} = \frac{1\ \text{m}}{343\ \frac{\text{m}}{\text{s}}}$$

$$\text{Tiempo de recorrido de onda reflejada} = 0.00291\ \text{s} = 29.1\ \text{us}$$

En la Anexo A se muestra el script de adquisición de datos y su tabla de descripción, en la que se consideró el tiempo de recorrido de onda reflejada de la ecuación 3.2 y las características físicas del tanque de llenado como la altura máxima del mismo, ya que el set point se va a configurar con centímetros de altura en el tanque de llenado.

En la Figura 3.12 se gráfica la reacción de la planta a la función escalón con respecto al tiempo, el nivel esta medido en centímetros, en la gráfica se puede observar el cambio de estado transitorio a estado estable.

Figura 3.12. Función de la respuesta de la planta la función escalón



Respuesta de la planta representada en gráfica de Excel, (Sánchez Bryan & Tupiza Alex)

3.6.2 Generación de función de transferencia de la planta en el software Matlab

Para generar la función de transferencia se utiliza un método experimental que identifica el modelo matemático de la planta, el cual busca describir el comportamiento de la misma. Para este fin se usa la herramienta "IDENT" de Matlab que se configura para lograr la identificación de forma exitosa, para este caso se identifica a un sistema de segundo orden.

El software Matlab puede importar cada uno de los 200 datos adquiridos en total con un tiempo de muestreo de 250 milisegundos, copiándolos directamente desde un archivo con extensión tipo .txt en un bloc de notas o tipo .xls en un libro de trabajo de Excel.

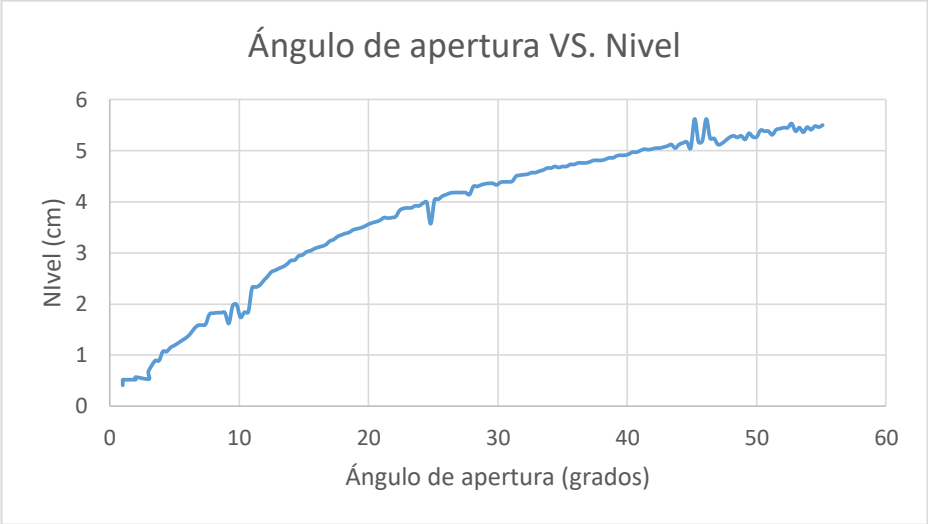
Automáticamente se muestra el contenido dentro del libro de trabajo de Excel, aquí se selecciona los datos que se van a importar a manera de variables como vector columna (Column vectors) ya que están dispuestos en las celdas verticalmente, como se muestra en la Figura B.1 del Anexo B.

El modelo matemático se obtiene identificando una entrada y una salida en el proceso, la entrada es la función escalón, que es el estímulo que se le da a la planta con el accionamiento de 55 grados de la electro-válvula.

La salida es el comportamiento que se refleja en el cambio de nivel en el tanque de llenado, gracias a la acción de la función escalón

En la Figura 3.13 se gráfica las funciones de entrada vs salida en Excel

Figura 3.13. Relación de apertura de válvula vs nivel en tanque de llenado



Gráfica de relación de los grados de apertura de válvula con el nivel del combustible en Excel, (Sánchez Bryan & Tupiza Alex)

El procesamiento del modelo matemático se realiza con el "IDENT" de Matlab, es una herramienta exclusiva para generar la función de transferencia a partir de datos reales mediante iteraciones asemeja el porcentaje del comportamiento experimental de la planta al modelo de salida que se obtiene con el IDENT.

La estimación de la función de transferencia se realiza con la opción "Process Models", en el modelamiento se configura los siguientes parámetros:

Poles: El número de polos que tiene la función de transferencia, en este caso se asignan 2 por que este se aproxima más el comportamiento de la planta.

Delay: El casillero delay se desactiva ya que la función escalón se aplica desde el tiempo cero, entonces no existe retardo.

Zero: Este casillero asigna zeros es el numerador de la función de transferencia.

Name: Asigna un nombre a la función de transferencia.

Todos los demás parámetros no se configuran, para que la herramienta “Process Model” comience con las iteraciones para definir el modelo matemático una vez seleccionada la opción “estimate”.

En la Figura B.2 del Anexo B está ilustrado el proceso en la herramienta “Process Model”, con las configuraciones correspondientes.

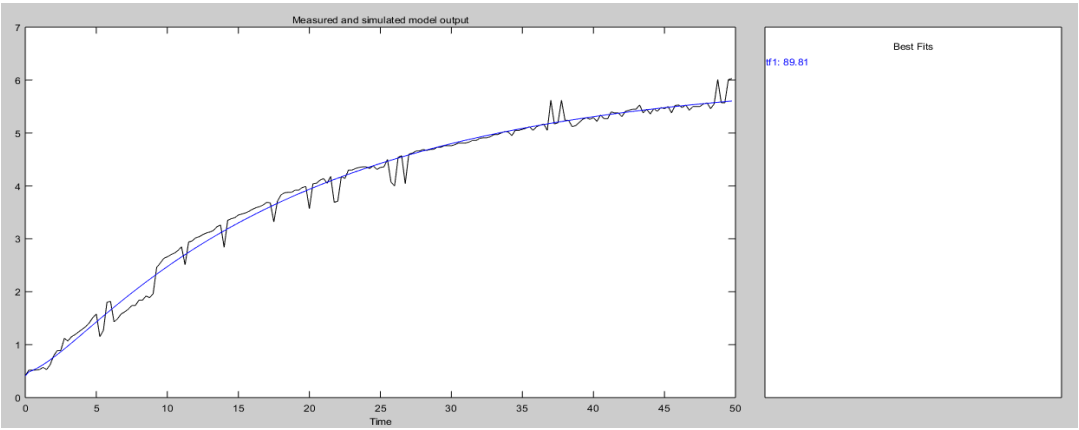
La función de transferencia es generada y se guarda en el “IDENT” con el nombre de “tf1”, para exportar tf1 al Workspace se arrastra el gráfico hacia el recuadro “To Workspace” como se indica en la Figura B.3 del Anexo B, en este también se muestran las características generales de tf1, estas se pueden apreciar ejecutando la variable tf1 en el panel Comand Window.

La función de transferencia tf1 tiene un porcentaje de coincidencia de 89.81%, consta de 2 polos y un zero, en la Figura 3.14 se muestra la gráfica del modelo de la salida en color azul y los datos reales en color negro.

La función de Transferencia obtenida de la planta se muestra en la ecuación 3.3

$$FT = \frac{-0.003263\ s + 0.003073}{s^2 + 0.583\ s + 0.02797} \qquad \text{Ec (3.3)}$$

Figura 3.14. Modelo Output de función de transferencia



Interfaz de IDENT de Matlab, (Sánchez Bryan & Tupiza Alex)

3.7 Diseño del controlador LQG

Para el desarrollo del controlador LQG, se necesita diseñar un estimador y un regulador, ambos valores se pueden calcular mediante las funciones LQE () y LQR () respectivamente con la ayuda del software Matlab.

Conociendo la función de transferencia de la planta la cual se mostró en la ecuación 3.3 se pueden obtener las matrices de estado A, B, C y D con la ayuda del comando *[A, B, C, D] = tf2ss (numerador, denominador),* como se muestra en la Figura 3.15.

Figura 3.15. Obtención de las matrices de estado A, B, C, D

```
%tf1
numerador =[0 -0.003263   0.003073]
denominador = [1 0.583   0.02797]
FT=tf(numerador,denominador)

% MATRICES DE ESTADO ABCD
[A,B,C,D]=tf2ss(numerador,denominador)
```

Variables de control LQG en Matlab, (Sánchez Bryan & Tupiza Alex)

Obteniendo así las matrices de estado, mostradas en las ecuaciones 3.4 a 3.7:

$$A=\begin{pmatrix}-0.5830 & -0.0280\\ 1.0000 & 0\end{pmatrix} \qquad \text{Ec. (3.4)}$$

$$B=\begin{pmatrix}1\\ 0\end{pmatrix} \qquad \text{Ec. (3.5)}$$

$$C=(-0.0033 \;\; 0.0031) \qquad \text{Ec. (3.6)}$$

$$D = 0 \qquad \text{Ec. (3.7)}$$

El regulador se puede calcular mediante la función *[K] = lqr(A, B, Q, R) se* puede obtener la matriz de ganancia K, mostrada en la ecuación 3.9, donde Ay B representan las matrices de estado, Q y R son matrices de función de coste que se calculan mediante simulación, dependiendo de los resultados dichos valores se modifican y se recalculan un cierto número de iteraciones hasta obtener un valor que el diseñador considere aceptable.

Como ayuda para el cálculo de las matrices, se puede escoger R como una matriz identidad del tamaño de la matriz de estado B, mientras que la matriz Q se puede obtener como se muestra en la ecuación 3.8

$$[Q]= [C]^{T}\text{-}[C] \qquad \text{Ec. (3.8)}$$

Con lo mencionado anteriormente el resultado de la matriz de ganancia K, es el siguiente:

$$K = (0.3019x10^{-3}\ 0.17x10^{-3}) \qquad \text{Ec. (3.9)}$$

Por su parte el estimador, se calcula mediante la función *[L]=lqe(A, G, C, W (i), V (i))*, donde A y C son matrices de estado, G, W y V son matrices de ruido ingresadas de manera aleatoria al sistema, obteniendo así el resultado mostrado en la ecuación 3.10.

$$L= \begin{pmatrix} -0.1135 \\ 2.7231 \end{pmatrix} \qquad \text{Ec. (3.10)}$$

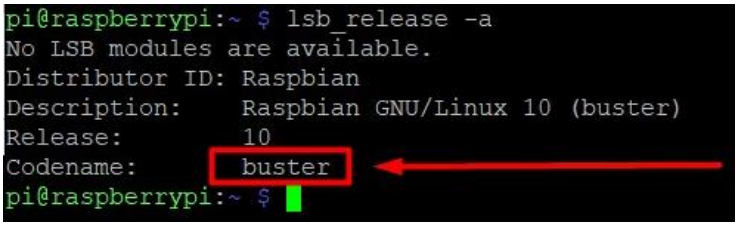
3.8 Instalación de MQTT broker (mosquitto) en la tarjeta Raspberry Pi 3 modelo B+

La tarjeta Raspberry funciona con el sistema operativo Raspbian, este sistema operativo no tiene instalado por defecto MQTT, este protocolo necesita un broker llamado mosquitto que es el que comunica la raspberry con los distintos dispositivos que se conectan por MQTT como la cámara web y el módulo ESP8266 NODE a nivel de hardware y software.

En la instalación del broker MQTT se distinguen los siguientes pasos:

- Comprobación de la versión de Raspbian: En el terminal de comandos se escribe el comando lsb_release –a, se despliega la información sobre la versión que tiene actualmente el dispositivo, como se muestra en la siguiente Figura 3.16, esta versión debe ser buster ya que es compatible para el broker a instalar.

Figura 3.16. Comprobación de versión en Raspbian



Terminal de comandos en Raspberry, (Sánchez Bryan & Tupiza Alex)

- Descarga del paquete mosquitto: En el repositorio de Raspbian se importa la llave de firma del paquete con el comando *wget http://repo.mosquitto.org/debian/mosquitto-repo.gpg.key* . Para almacenar el archivo en un directorio actual se usa el comando *cd /etc/apt/sources.list.d/*.
-
- El siguiente paso es instalar el paquete de datos según la versión de Raspbian en la dirección escogida anteriormente con el comando *sudo wget http://repo.mosquitto.org/debian/mosquitto-buster.list* .

  El siguiente paso es actualizar el sistema con el comando sudo apt-get update.

- Cargar MQTT en la tarjeta Raspberry: El protocolo MQTT puede reconocer a la raspberry como cliente o como servidor, el broker mosquitto hace las veces de servidor a la vez que necesita reconocer los clientes con los que interactúa. El comando sudo apt install mosquitto mosquitto-clients sirve para instalar los paquetes de clientes.

3.9 Instalación de Node Red en la tarjeta Raspberry Pi 3 modelo B+

Node red es la herramienta gráfica de programación basada en Node.js, este software permite crear un flujo llamado “flow1” que ordenada lógicamente los nodos para comunicar inalámbricamente los clientes suscritos al mismo topic, siendo los clientes la cámara web y el módulo ESP8266 NODE, para implementar el control LQG.

Para instalar el software Node Red la tarjeta raspberry debe tener conexión a internet y se realiza los siguientes pasos:

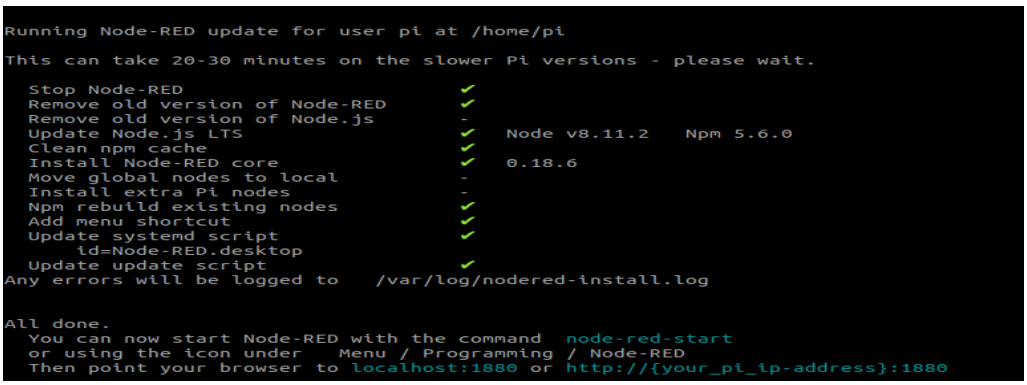
- Actualización de paquetes: La actualización de los paquetes y las versiones de los mismos se realiza con el comando sudo apt-get update

- Instalación de paquetes actualizados: El comando sudo apt-get upgrade instala la última versión de los paquetes disponibles.

- Instalación de Node Red: Node Red esta preinstalado en el sistema operativo Raspbian pero con una versión antigua de Node.js, para evitar problemas de ejecución se actualiza la versión con el comando bash < (curl -sL https://raw.githubusercontent.com/node-red/raspbian-deb-package/master/resources/update-nodejs-and-nodered).

  Con el comando de actualización aparece una pregunta de confirmación, en la cual se escribe la palabra Yes. Consecuentemente aparece una ventana de verificación que corre el software Node Red e indica los paquetes instalados con un visto tal y cómo se puede apreciar en la Figura 3.17.

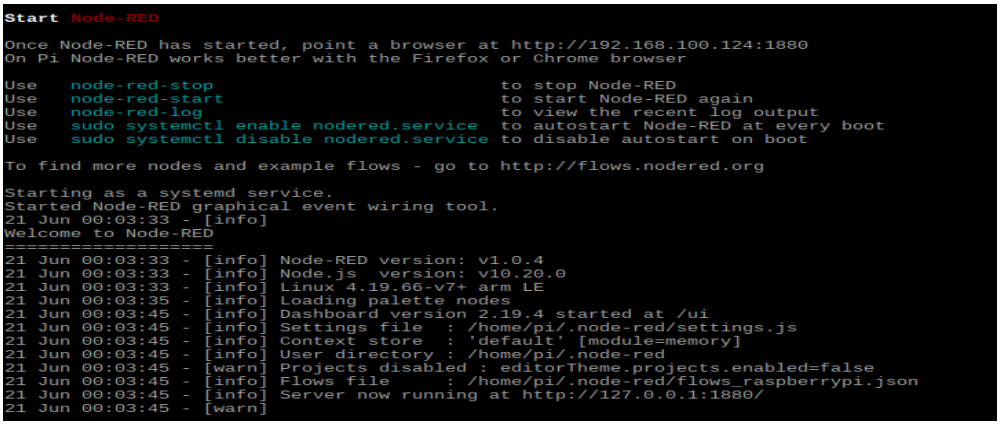
Figura 3.17. Prueba de estado de MQTT



Terminal de comandos en Raspberry, (Sánchez Bryan & Tupiza Alex)

En la Figura 3.18, se muestra como Node Red se ejecuta y se despliegan los registros del terminal, conociendo la Dirección IP del servidor y al puerto que apunta, en este caso 1880.

Figura 3.18. Inicialización de Node Red

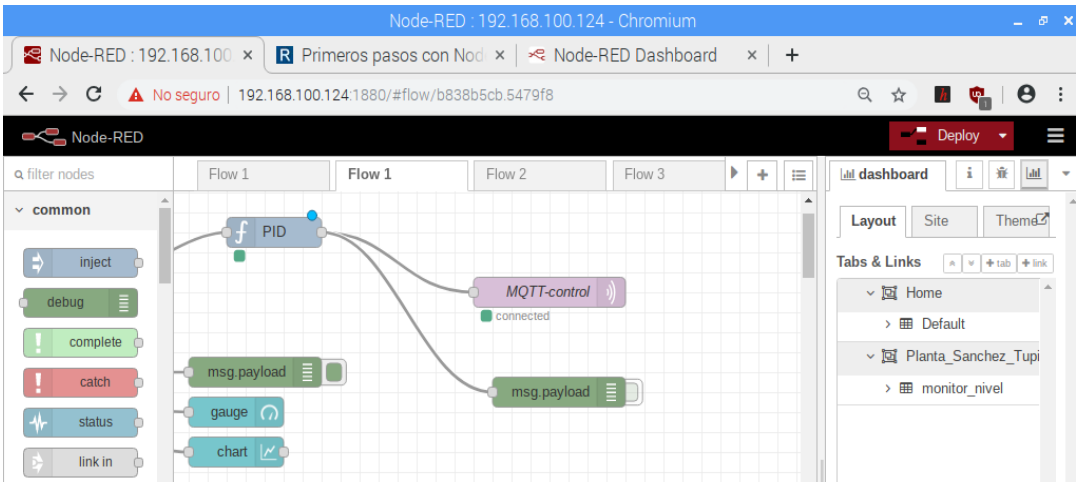


Terminal de comandos en Node Red, (Sánchez Bryan & Tupiza Alex)

La interfaz de Node Red se visualiza escribiendo la dirección URL donde inicia el servidor en el buscador que disponga cualquier dispositivo conectado a la red local donde se ejecuta Node Red.

En la Figura 3.19 se visualiza el entorno y las diferentes funciones de Node Red con el buscador apuntando hacia la dirección IP del servidor.

Figura 3.19. Entorno de Node Red



Interfaz de Node Red, (Sánchez Bryan & Tupiza Alex)

3.10 Implementación de sensado por visión artificial

La visión artificial funciona básicamente con la librería OpenCV y Numpy, la adquisición de imágenes se realizó con una cámara web. El estado de nivel de la planta es sensado en tiempo real por un algoritmo desarrollado en Python que obedece el diagrama de bloques de la Figura 3.20

Figura 3.20. Diagrama de bloques de visión artificial

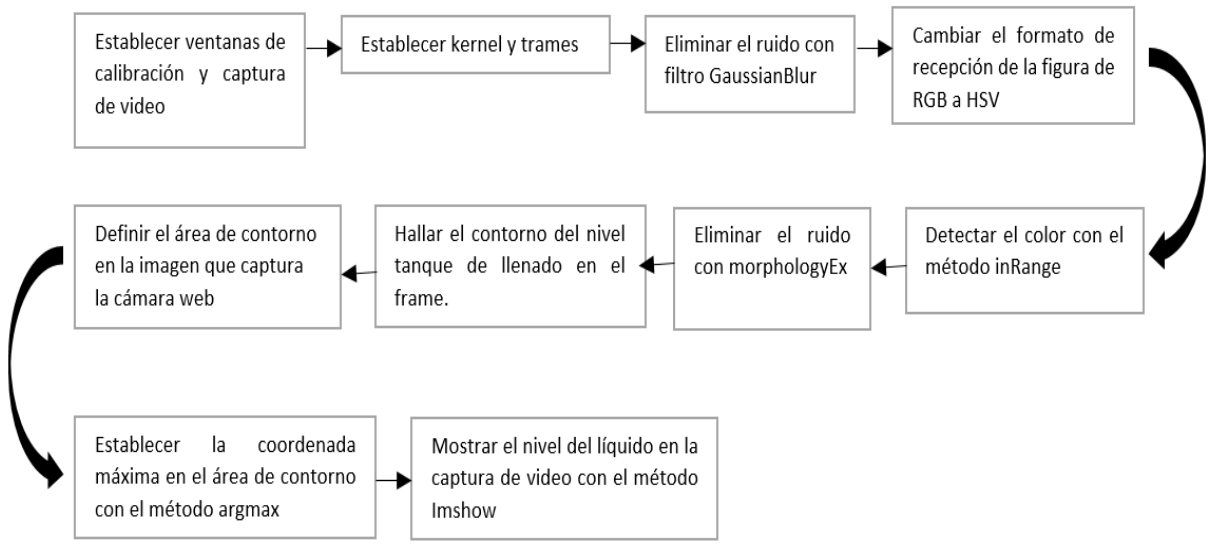


Diagrama de bloques de visión artificial, (Sánchez Bryan & Tupiza Alex)

Las palabras desconocidas de visión artificial están explicadas en los siguientes conceptos:

- Kernel: Es la matriz de convolución que realiza las transformaciones a nivel morfológico de las imágenes.

- Frame: son las imágenes captadas por la cámara web, se puede extraer la información de nivel desde los frames.

- BGR: Es un formato que representa a las imágenes en una combinación que utiliza únicamente tres colores (verde, azul y rojo).

- HSV: Es un formato que representa a las imágenes con tres parámetros (saturación matiz y brillo).

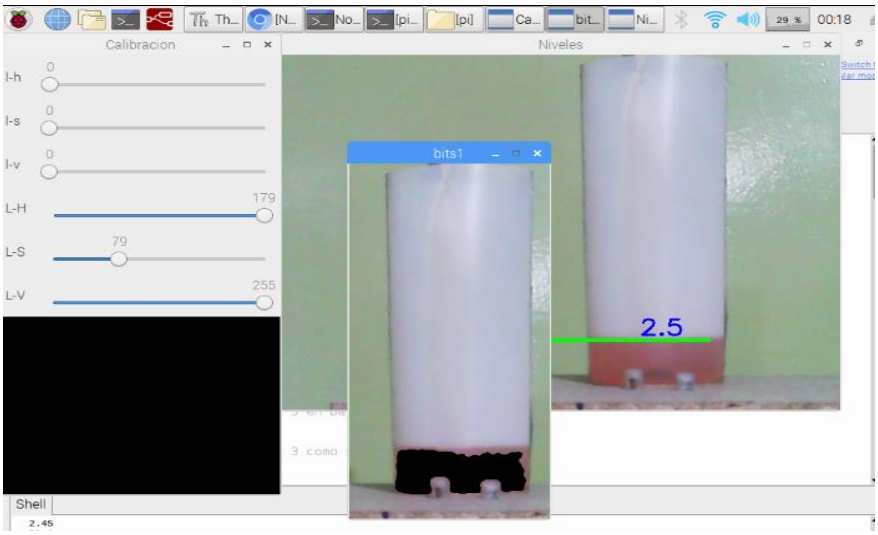
- Contorno: Es la figura de datos binarios que se forma por pixeles del mismo color.

- El algoritmo de visión artificial está desarrollado en el IDE Thonny de Python, este algoritmo refleja las acciones del diagrama de bloques para que el sistema tenga noción del nivel de combustible en el tanque de llenado.

En el Anexo C se muestra el algoritmo de visión artificial y la descripción del mismo.

En la Figura 3.21 se puede apreciar el interfaz resultante del programa de visión artificial con cada una de sus ventanas desplegadas.

Figura 3.21. Ejecución de programa de visión artificial



Ventanas de calibración y adquisición de imágenes para el sensado del nivel de combustible, (Sánchez Bryan & Tupiza Alex)

3.11 Desarrollo de accionamiento inalámbrico con el módulo ESP8266 NODE

EL módulo ESP8266 se conecta inalámbricamente a la red local mediante la antena Wi-Fi que posee el dispositivo, la conectividad es compatible lógicamente con el protocolo TCP/IP pero para que el módulo sea reconocido en el puerto COM del computador se instala el driver CH341SE para placas genéricas.

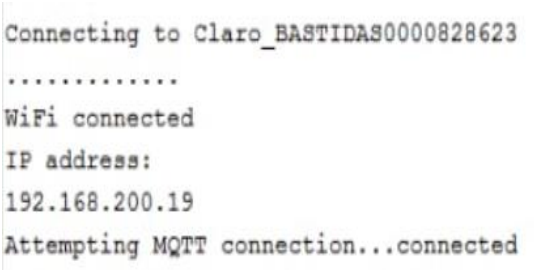
La ejecución del instalador se realiza estableciendo como administrador al setup.exe que existe dentro en el archivo ZIP comprimido, consecuentemente aparece el interfaz de instalación como se muestra en la Figura D.1 del Anexo D.

La conectividad del módulo ESP8266 se desarrolla en el mismo IDE de Arduino y se necesitan varios datos específicos para que la red local reconozca al dispositivo.

En la Figura D.2 del Anexo D se muestra el algoritmo y una tabla con la descripción de sus funciones que se utiliza en el módulo para la conectividad y el accionamiento de la servo-válvula.

En la Figura 3.22 se muestra los datos de conexión del módulo en el monitor serial de Arduino.

Figura 3.22 Datos de conexión Wi-Fi en el monitor serial de ESP8266.



Puerto serial de Arduino, (Sánchez Bryan & Tupiza Alex)

3.12 Desarrollo de Monitoreo con Telegram

Como primer paso para tener la comunicación entre el software de Telegram y la raspberry se tiene que instalar una serie de librerías, para realizar la instalación se accede al terminal de la raspberry, y se tiene que actualizar el sistema mediante el comando:

sudo apt-get update && sudo apt-get upgrade –y

Como siguiente punto se instalan los paquetes necesarios para el funcionamiento del programa, los cuales son:

sudo apt-get install python-setuptools

sudo apt-get install python-pip

sudo pip install telepot

sudo apt-get install sysstat

Mediante el comando Git, se clona al repositorio de la API de Telegram.

sudo git clone https://github.com/eternnoir/pyTelegramBotAPI.git /opt/pyTelegramApi

Finalmente se instalan los paquetes mediante los siguientes comandos:

cd /opt/pyTelegramBotApi

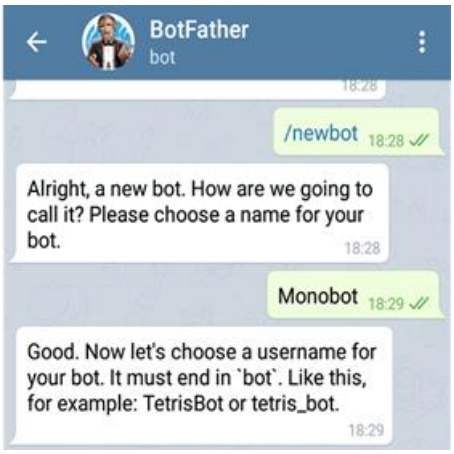
sudo python setup.py install

3.12.1 Creación e implementación del bot

Una vez realizados los pasos de instalación en el servidor, se procede a descargar la aplicación de Telegram en un dispositivo móvil, y se instala ingresando un número telefónico, el nombre y apellido para el usuario.

Una vez dentro de la aplicación se busca la opción BotFather, y dentro del chat se selecciona la opción INICIAR y se escribe el comando */newbot*, posteriormente se le indicará a la aplicación el nombre del bot, como se muestra en la Figura 3.23

Figura 3.23. Creación del bot e indicación del nombre mediante BotFather



Interfaz de Telegram, (Sánchez Bryan & Tupiza Alex)

Cuando esté listo el programa mostrará un mensaje indicando que el bot está creado, y se desplegará una lista de acciones que se pueden realizar, además la aplicación enviará el token perteneciente al bot para utilizar la API de Telegram.

Para dar un listado de acciones que el bot realizará, se ingresa al chat con botfather nuevamente, se escoge al bot creado y se ingresan las acciones a realizar mediante el comando /setcommands como se muestra en la Figura 3.24.

Figura 3.24. Comandos configurados en el bot

Interfaz de Telegram, (Sánchez Bryan & Tupiza Alex)

3.12.2 Comunicación Telegram-Python

Para la programación en python se importan las librerías Telebot instalada previamente desde el terminal, luego se ingresa el TOKEN recibido al crear el bot, para poder obtener la comunicación entre raspberry y Telegram como se muestra en la Figura 3.25.

Figura 3.25. Configuración de parámetros iniciales del bot.

```
import RPi.GPIO as GPIO #Importar la libreria para habil
import cv2 as cv        #Importar la libreria OpenCV par
import numpy as np      #Importar la libreria Numpy para
import telebot
from telebot import types

TOKEN = '849653296:AAHQaGmDuTspHDkstl7HbbyW0_fPELUsswo'
bot = telebot.TeleBot(TOKEN)
```

Parámetros principales de la librería paho.mqtt, (Sánchez Bryan & Tupiza Alex)

Dentro del script de Python se pueden agregar condiciones para el monitoreo de nivel, así como las distintas acciones que el bot realizará con los comandos ingresados, tales como el dato leído por la cámara, el porcentaje de nivel, que puede ser bajo, medio o alto y una foto del nivel en tiempo real como se muestra en la Figura 3.26 a continuación:

Figura 3.26. Configuración de acciones del bot.

```
@bot.message_handler(commands=['nivel'])
def comando_nivel(mensaje):
        chat_id = mensaje.chat.id
        bot.send_message(chat_id, 'EL NIVEL SE ENCUENTRA EN :')
        bot.send_message(chat_id, nivel_porcentaje1 )
        cv.imwrite("foto.png",frame)
        photo =open('/home/pi/Desktop/foto.png','rb')
        bot.send_photo(chat_id,photo)
```

Parámetros principales del bot de Telegram, (Sánchez Bryan & Tupiza Alex)

Cuando se envíe el comando **/nivel** previamente configurado en la aplicación, el bot enviará un mensaje de vuelta con el texto “EL NIVEL SE ENCUENTRA EN:”, junto con el dato de la variable nivel_porcentaje1 que contiene el valor entero del nivel obtenido por la cámara web, y con la ayuda de la librería cv se enviará una foto en el momento que el comando /nivel se ejecute. El resultado se muestra a continuación en la Figura 3.27:

Figura 3.27. Prueba de comunicación Telegram- Raspberry

Interfaz de Telegram, (Sánchez Bryan & Tupiza Alex)

El funcionamiento de la aplicación Telegram con el algoritmo de visión artificial se muestra en la Figura 3.28, en el siguiente diagrama de flujos.

Figura 3.28. Diagrama de flujo del algoritmo de Telegram

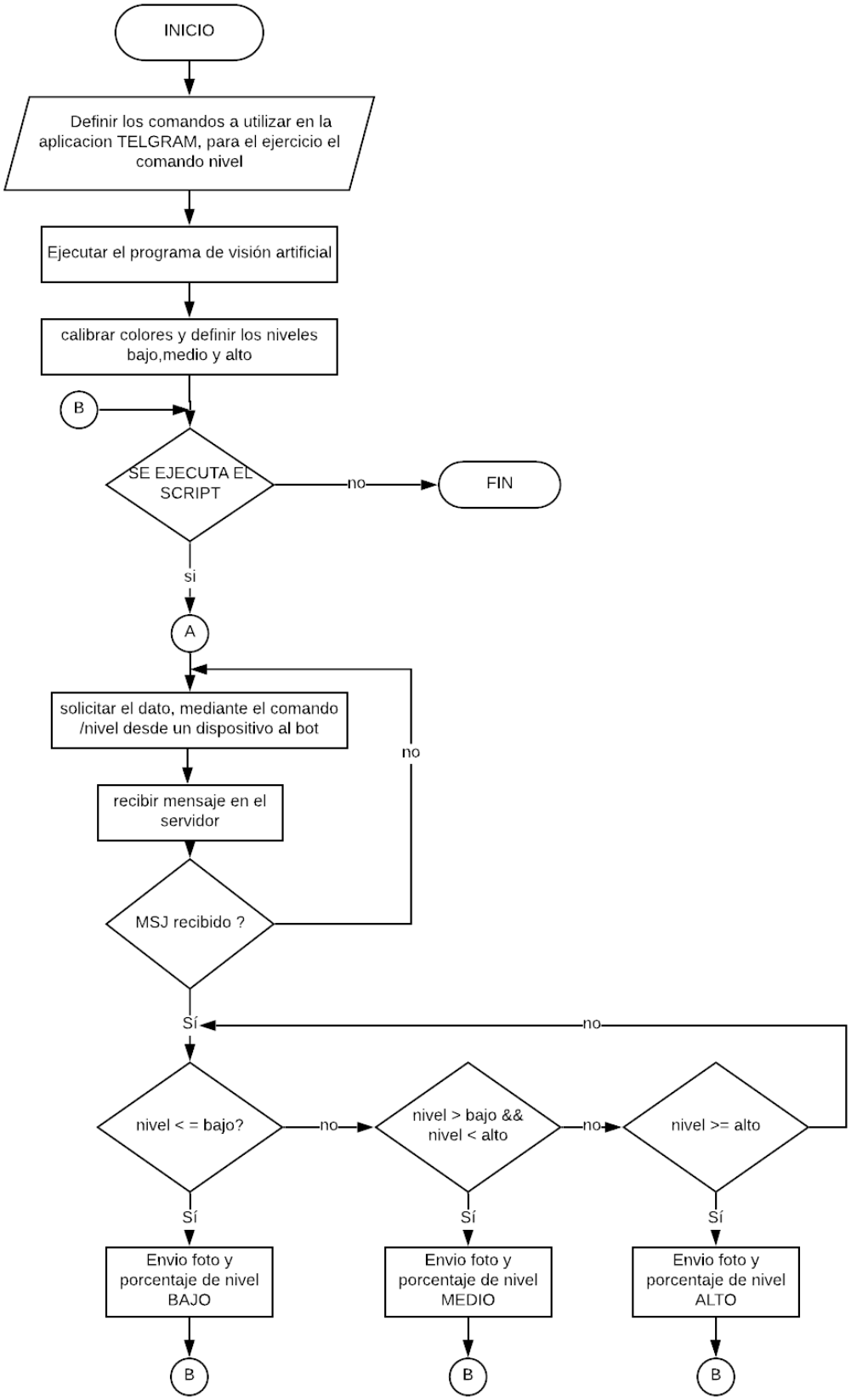
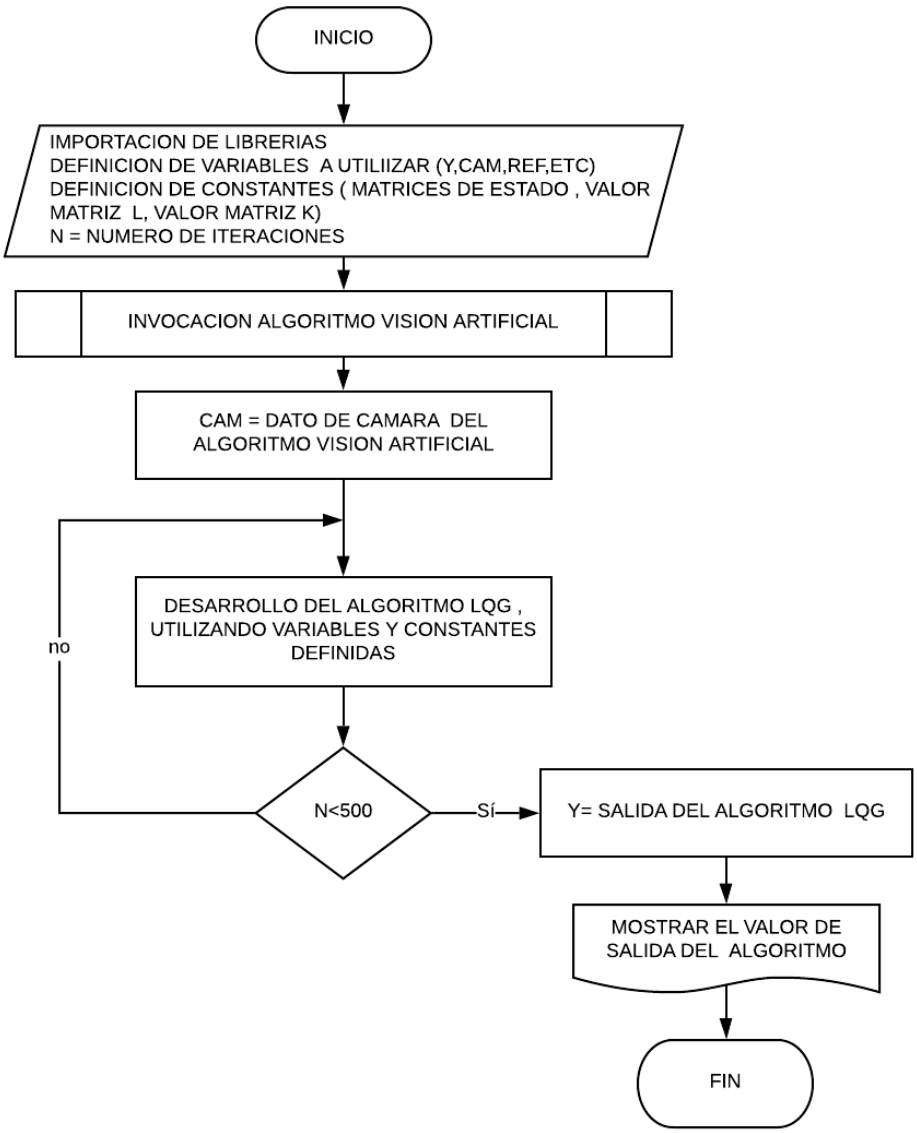


Diagrama de flujo de monitoreo remoto, (Sánchez Bryan & Tupiza Alex)

3.13 Incorporación de monitoreo remoto con control LQG en Node Red

El funcionamiento del algoritmo LQG desarrollado en Python se muestra a continuación en la Figura 3.29

Figura 3.29. Diagrama de flujo de algoritmo LQG en python



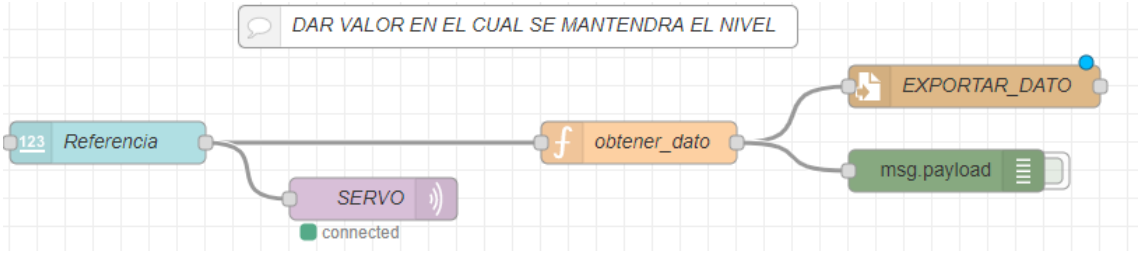
Funcionamiento explicado del controlador LQG, (Sánchez Bryan & Tupiza Alex)

El script desarrollado en Python trata de aproximar la salida del controlador LQG al dato de cámara obtenido por el algoritmo de visión artificial, realizando un determinado número de iteraciones, donde se utilizan constantes tales como las matrices de estado, los valores de las matrices K y L previamente obtenidas en el software Matlab, así como una serie de variables donde se almacenarán los resultados

obtenidos al realizar las distintas operaciones matemáticas, y con el fin de que la salida del algoritmo pueda ser invocada desde Node Red para poder realizar futuras comparaciones al enviar el dato de salida al software de Arduino.

El desarrollo del flujo programado en Node Red se muestra a continuación en la Figura 3.30.

Figura 3.30. Flujo en Node Red para lectura de cámara



Flujo principal de Node Red, (Sánchez Bryan & Tupiza Alex)

La descripción de los nodos de la Figura 3.30 se muestra a continuación en la Tabla 3.7.

Tabla 3.7. Descripción de las funciones de los nodos de lectura de cámara.

Nodo	Función
Referencia	Permite ingresar un valor entre 0 y 20 cm, con la finalidad de que el líquido se mantenga en el valor ingresado.
Servo	Envía el dato de referencia al software Arduino, mediante el protocolo MQTT.
Obtener_dato	Toma el dato ingresado en el nodo de referencia.
Exportar_dato	Toma el dato de referencia y lo guarda en un archivo .txt para que pueda ser llamada desde cualquier script de Python.
msg payload	Permite visualizar el valor desde la consola de Node Red.

Tabla de descripción de los nodos en el flujo de lectura de cámara en Node Red, (Sánchez Bryan & Tupiza Alex)

Para la lectura del dato de cámara es necesario el uso de la librería paho.mqtt la cual permite al script de visión artificial desarrollado en python funcionar como un cliente del protocolo mqtt, para ello se debe importar las librerías necesarias desde el script como se muestra en la Figura 3.31.

Figura 3.31. Flujo en Node Red para lectura de cámara

```
import paho.mqtt.publish as publish
import paho.mqtt.client as mqtt
broker_address="192.168.0.102"

client = mqtt.Client("P1") #create new instance
client.connect(broker_address) #connect to broker
```

Parámetros principales para utilización de librería paho.mqtt, (Sánchez Bryan & Tupiza Alex)

En la Figura 3.32 se muestran los comandos con los que se puede enviar el dato a Node Red, se utiliza el método client.publish.

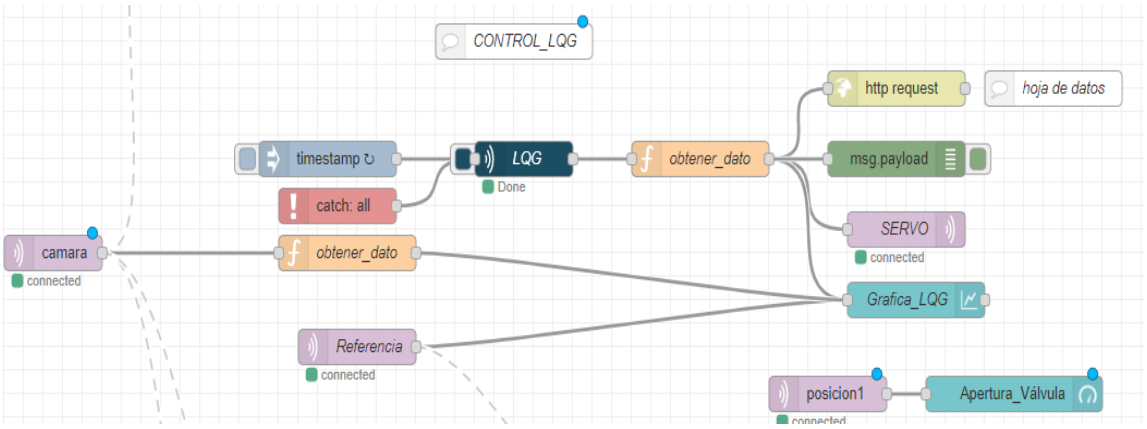
Figura 3.32 Función de paho.mqtt para llamar al dato nivel en Node Red

```
client.publish("camara",nivel_porcentaje1)#publish
client.publish("referencia",ref)#publish
```

Parámetros de la librería paho.mqtt para el envio de datos a Node red (Sánchez Bryan & Tupiza Alex)

El control LQG y la visualización en el dashboard se integran en Node Red como se muestra en la Figura 3.33

Figura 3.33. Flujo en Node Red con funcionalidad del dashboard y control LQG



Flujo principal de Node Red, (Sánchez Bryan & Tupiza Alex)

La descripción de nodos de la Figura 3.33 se muestra a continuación en la Tabla 3.8.

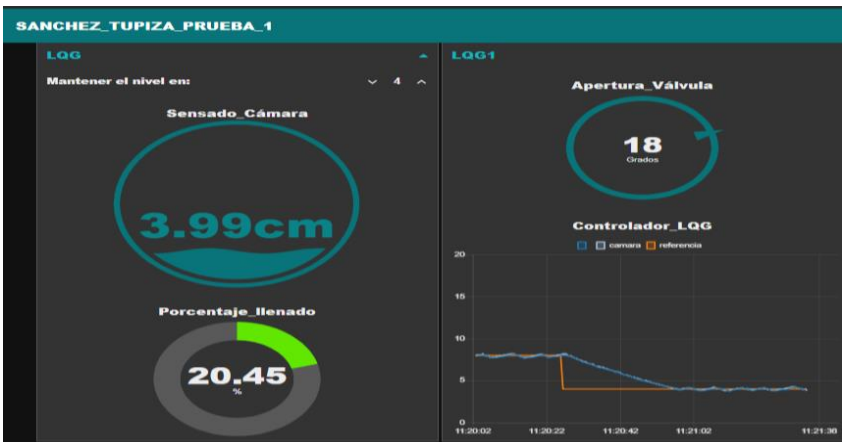
Tabla 3.8. Descripción del funcionamiento de los nodos para el control LQG

Nodo	Función
Cámara	Lee el dato de cámara proveniente del algoritmo de visión artificial
timestamp	Permite ingresar un intervalo de tiempo en segundos para repetir una acción dentro de un nodo
catch all	Permite captar cualquier error que pueda producirse dentro de algún nodo
obtener_dato	Toma el dato de cámara y lo convierte a tipo String de tal manera que otros nodos puedan trabajar con el
lqg	Ejecuta el algoritmo LQG desarrollado en el script de Python
http request	Envía los datos obtenidos del script a una hoja de cálculo en Google Drive
msg payload	Permite visualizar el valor desde la consola de Node Red
servo	Envía el dato de salida del script al software Arduino
referencia	Toma el dato de la referencia de Node Red
Gráfica lqg	Muestra una gráfica en el dashboard con los valores de la referencia, salida del LQG, y cámara
posicion1	Toma el dato de la posición del servo motor desde el software Arduino
apertura _ válvula	Muestra el valor de la posición del servo motor en el dashboard

Tabla de descripción de los nodos en el flujo para incorporar el control LQG en Node Red, (Sánchez Bryan & Tupiza Alex)

Con lo mencionado anteriormente el resultado en el dashboard se muestra en la Figura 3.34

Figura 3.34. Dashboard del control LQG inalámbrico de nivel en Node Red



Dashboard principal en Node Red, (Sánchez Bryan & Tupiza Alex)

CAPÍTULO 4

PRUEBAS Y RESULTADOS

En este capítulo se detallan las distintas pruebas a realizar, con el fin de obtener un óptimo funcionamiento de la planta, además de los resultados obtenidos al comparar tanto el controlador PID con el controlador LQG

4.1 Prueba de selección del material a utilizar como tanque de llenado

En las primeras pruebas se utilizó una botella de plástico transparente como se muestra Figura 4.1.

Figura 4.1. Prueba de monitoreo botella de plástico

Pruebas con tanque de llenado de plástico, (Sánchez Bryan & Tupiza Alex)

Se puede apreciar que el valor entregado en mensaje por el bot difiere en un mínimo porcentaje de la foto recibida, esto se debe al brillo generado por la luz y para la cámara web resulta complicado realizar un sensado adecuado.

Con el fin de corregir este problema, en la Figura 4.2 se utilizó una probeta de vidrio con poca luz y un fondo blanco, mientras que en la Figura 4.3 se utilizó un recipiente de plástico no transparente de color blanco, con una mayor iluminación.

Figura 4.2. Prueba de monitoreo probeta de vidrio

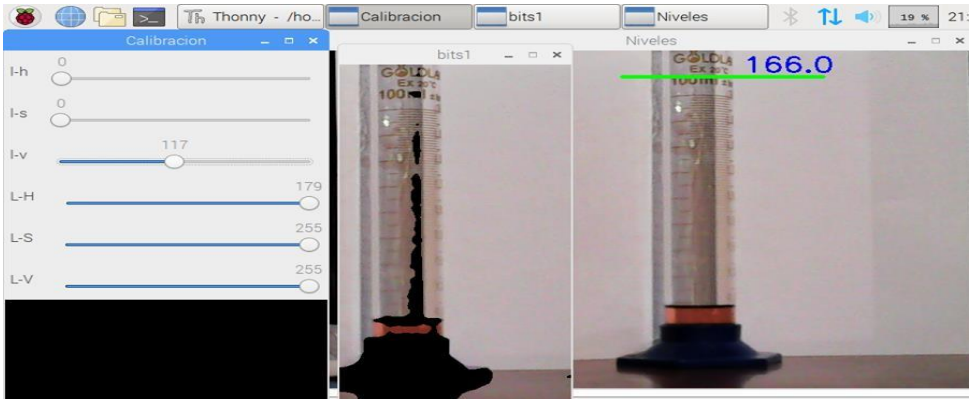
Pruebas con tanque de llenado de cristal, (Sánchez Bryan & Tupiza Alex)

Figura 4.3. Prueba de monitoreo botella de plástico no transparente

Pruebas con tanque de llenado de plástico opaco, (Sánchez Bryan & Tupiza Alex)

Como se puede observar en ambas figuras el sensado mejora, sin embargo, en la probeta de vidrio se generaban problemas con el contraste cuando se medían valores de nivel alto, ya que parte del líquido quedaba como residuo en la probeta cuando el líquido bajaba a un nivel normal y a un nivel bajo, como se muestra en la Figura 4.4

Figura 4.4. Prueba de monitoreo probeta de vidrio



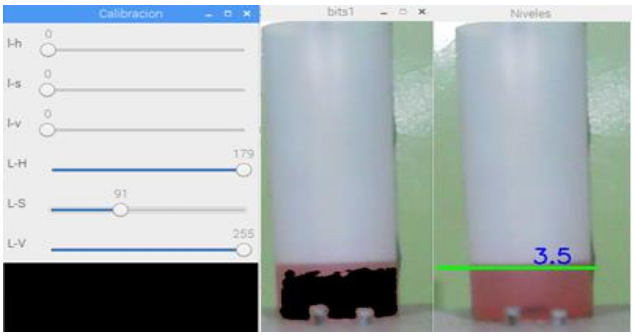
Pruebas con tanque de llenado de cristal, (Sánchez Bryan & Tupiza Alex)

Con el fin de evitar este error y para efecto de pruebas lo conveniente era secar la probeta, así que finalmente se procedió a utilizar la probeta de plástico no transparente de color blanco con el fin de evitar problemas de brillo, reflexión de luz y líquidos residuales que permanezcan alojados en el recipiente, ya que de esta manera la cámara se enfoca únicamente en el color del líquido haciendo más fácil su sensado.

4.2 Prueba de exactitud del sensado

Para determinar qué tan exactos eran los valores recibidos por el sistema implementado se tomaron 20 datos en un mismo nivel (3,5 cm), como se muestra en la Figura 4.5, obteniendo los siguientes resultados.

Figura 4.5. Valor de nivel para realizar pruebas de exactitud en las medidas



Pruebas con tanque de llenado de plástico, (Sánchez Bryan & Tupiza Alex)

Los distintos errores presentes en las medidas se muestran en la Tabla 4.1, y se pudieron calcular mediante el uso de la ecuación 4.1 que representa el error absoluto y la ecuación 4.2 que representa el error relativo.

$$Ea=\overline{X}-Xi \qquad \text{Ec. (4.1)}$$

Donde ($\bar{X}$) es el valor real de la medida (3,5 cm) y (Xi) el valor que se ha obtenido en la medición.

$$Er=\left(\frac{Ea}{\overline{X}}\right).100\% \qquad \text{Ec. (4.2)}$$

Los valores obtenidos al aplicar las fórmulas de error se pueden observar en la tabla E.1 del ANEXO E.

Tabla 4.1. Resultados de error obtenido de 20 datos a un nivel de 3.5 cm

ERROR ABSOLUTO	ERROR RELATIVO
-0,029 cm	0,829%

Resultados al aplicar las ecuaciones de error relativo y absoluto, (Sánchez Bryan & Tupiza Alex)

Como se puede apreciar en la Tabla 4.1 el promedio de errores es de 0,83% aproximadamente, lo que muestra que el sistema trabaja de una manera eficiente.

4.3 Prueba de control de nivel PID

Para las pruebas con el controlador PID se variaron los intervalos de nivel, y en cada uno de ellos se tomó un dato cada 0,25 segundos, para almacenar los datos y analizarlos se utilizó el nodo http request de Node Red, que envía el dato de salida del controlador a una hoja de cálculo en Google drive, permitiendo obtener mediante una gráfica en Excel como se muestra en la Figura 4.6, el tiempo de asentamiento, el tiempo de subida y el máximo sobre impulso.

Figura 4.6. Gráfica en Excel para el controlador PID, con el nivel requerido en 12cm

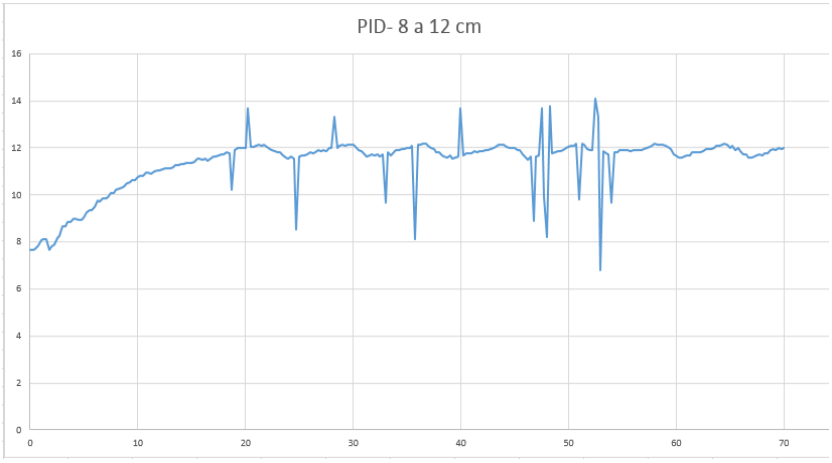


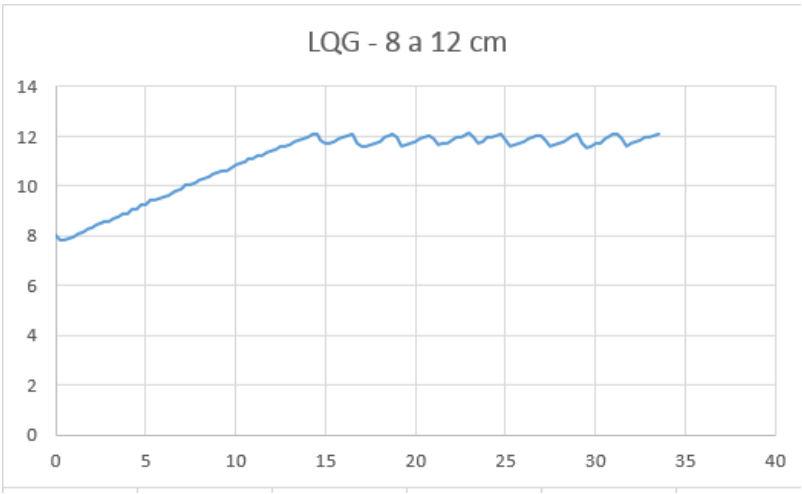
Gráfico en tiempo real del Control PID en el Excel, (Sánchez Bryan & Tupiza Alex)

Este mismo proceso se realizó en los distintos niveles, realizando así 25 pruebas diferentes como se muestra en la Tabla E.2, del anexo E.

4.4 Prueba de control de nivel LQG

Para las pruebas con el controlador LQG, se realizó el mismo proceso que con el controlador PID. Se realizaron 25 pruebas, variando los intervalos de nivel, en cada uno de los niveles cada 0,25 segundos, obteniendo así la gráfica mostradas en la Figura 4.7. Los resultados, se muestran en la Tabla E.3 del anexo E.

Figura 4.7. Gráfica en Excel para el controlador LQG, en 12cm



Datos experimentales del Control PID, (Sánchez Bryan & Tupiza Alex)

4.5 Comparación de medición entre controladores

Para esta prueba se calculó el error relativo y absoluto de medición con 30 datos en los niveles de 4cm, 8cm, 12cm, 16cm y 18cm, el análisis de las tablas se puede apreciar en las tablas E.2 y E.3 del Anexo E, teniendo así los siguientes resultados mostrados en las tablas 4.2 y 4.3

Tabla 4.2. Error absoluto y Error relativo en las mediciones realizadas por el controlador PID

CONTROL PID					
NIVEL	4cm	8cm	12cm	16cm	18cm
E. ABSOLUTO [cm]	0,13	0,02	0,05	0,17	0,12
E. RELATIVO [%]	4,69	1,86	1,37	1,48	0,86

Tabla de errores de control PID, (Sánchez Bryan & Tupiza Alex)

Tabla 4.3. Error absoluto y Error relativo en las mediciones realizadas por el controlador LQG

CONTROL LQG					
NIVEL	4cm	8cm	12cm	16cm	18cm
E. ABSOLUTO [cm]	0,06	0,02	0,02	0,06	0,10
E. RELATIVO [%]	2,62	1,03	0,97	0,81	0,74

Tabla de errores de control LQG, (Sánchez Bryan & Tupiza Alex)

Tal y como se muestra en las tablas anteriores, el controlador LQG tiene un error más pequeño en la medición y toma de datos respecto al controlador PID

4.6 Comparación de controladores con test de Wilcoxon

Las pruebas de control arrojan muestras aleatorias para testear una hipótesis nula, la hipótesis nula plantea que ambos escenarios son idénticas, es decir la prueba de Wilcoxon detecta diferencias entre los datos del control PID y los datos del control LQG. De manera intuitiva se puede imaginar en combinar dos muestras, asignar rangos a los datos medibles y definir al test de Wilcoxon como la adición de los rangos de las muestras medibles en uno de los dos controles. Si la suma es excesivamente pequeña

o excesivamente grande, es claro que los valores de un grupo de datos se consideran más pequeños o más grandes entre si y se rechazaría la hipótesis nula, al rechazar la misma quiere decir que existe una diferencia clara entre los tipos de control, demostrando cuál de ellos es el más óptimo.

El tiempo de subida es analizado en la Tabla E.4, el tiempo de estabilización en la Tabla E.5 y el máximo sobre impulso en la Tabla E.6, del anexo E. En cada una de las tablas se muestra la diferencia absoluta y el ranking de cada uno de los datos ordenándolos de tal manera que se diferencia los rangos negativos y positivos.

Las operaciones de la prueba de Wilcoxon respecto al tiempo de subida se pueden apreciar en la Tabla 4.4.

Tabla 4.4. Resultado de Prueba de Wilcoxon sobre el tiempo de subida.

Suma de Rangos positivos	0
Suma de Rangos negativos	325
Población N	25
Valor W	0
Valor Crítico	89

Resultados de la prueba de Wilcoxon para el tiempo de subida, (Sánchez Bryan & Tupiza Alex)

La comparación del valor W con el valor crítico, es el criterio que nos permite comparar al tiempo de subida en ambos tipos de control teniendo en cuenta lo siguientes parámetros.

Hipótesis nula (H0): No existe diferencia respecto al tiempo de subida con ambos tipos de control.

Si el valor W es menor al valor crítico se aprueba la Hipótesis nula, pero si en un caso W es mayor al valor crítico se rechazará la Hipótesis nula. En este proyecto se rechaza la Hipótesis nula; esto quiere decir que la planta si reacciona con un distinto tiempo de subida cuando se aplica el control LQG.

Las operaciones de la prueba de Wilcoxon respecto al tiempo de estabilización se pueden apreciar a continuación en la Tabla 4.5.

Tabla 4.5 Resultado de Prueba de Wilcoxon sobre el tiempo de estabilización.

Suma de Rangos positivos	0
Suma de Rangos negativos	325
Población N	25
Valor W	0
Valor Crítico	89

Resultados de la prueba de Wilcoxon para tiempo de estabilización, (Sánchez Bryan & Tupiza Alex)

Hipótesis nula (H0): No existe diferencia respecto al tiempo de estabilización con ambos tipos de control.

Si el valor W es menor al valor crítico se aprueba la Hipótesis nula, pero si en un caso W es mayor al valor crítico se rechazará la Hipótesis nula. En este proyecto se rechaza la Hipótesis nula; esto quiere decir que la planta si reacciona con un distinto tiempo de estabilización cuando se aplica el control LQG.

Los resultados de la prueba de Wilcoxon respecto al máximo sobre impulso se pueden apreciar a continuación en la Tabla 4.6.

Tabla 4.6. Resultado de Prueba de Wilcoxon sobre el máximo sobre impulso

Suma de Rangos positivos	310
Suma de Rangos negativos	15
Población N	25
Valor W	15
Valor Crítico	89

Resultados de la prueba de Wilcoxon para el máximo sobre impulso, (Sánchez Bryan & Tupiza Alex)

Hipótesis nula (H0): No existe diferencia respecto al máximo sobre impulso con ambos tipos de control.

Si el valor W es menor al valor crítico se aprueba la Hipótesis nula, pero si el valor W es mayor al valor crítico se rechaza la Hipótesis nula. En este caso se rechaza la Hipótesis nula; esto quiere decir que la planta si reacciona con un distinto máximo sobre impulso cuando se aplica el control LQG.

CONCLUSIONES

Para definir la función de transferencia de la planta se utilizó el método de la caja negra, este método permite que las características físicas de la planta describan perfectamente el comportamiento del sistema pues, al realizar varias pruebas se pudo constatar que esta opción de modelamiento matemático experimental es la mejor frente a un modelamiento teórico.

Con el uso de los nodos presentes en la programación gráfica de Node Red se pudo implementar el protocolo MQTT para emparejar al bróker alojado en la Raspberry Pi con los distintos suscriptores como son el algoritmo de sensado mediante visión artificial, los controladores y el programa desarrollado en el módulo ESP de Arduino, de tal manera que todos pudieron comunicarse inalámbricamente.

Al implementar el sistema de monitoreo remoto mediante la aplicación de Telegram, acciones que requieren estar cerca de la planta para supervisar el proceso dejan de ser tan necesarios, ya que mediante la aplicación y con la ayuda del bot creado se pudo observar el intercambio de información con los datos obtenidos por la cámara, este proceso se realiza mediante un mensaje de texto con el dato de cámara, adjuntando una foto del proceso que se está realizando, en tiempo real , haciendo que el monitoreo sea más fácil de realizar y desde cualquier lugar con una conexión a internet.

Con las distintas pruebas realizadas con los controladores de manera remota se pudo apreciar la diferencia que existe entre los mismos, siendo el LQG más rápido en tiempos de estabilización, y asentamiento así como también con un sobre impulso más pequeño mejorando el funcionamiento de la planta y optimizando al proceso ya que el controlador realiza un determinado número de iteraciones por segundo, aproximando el dato de salida al dato real, esto se puede apreciar con el error generado por el controlador LQG que resulta ser mínimo.

RECOMENDACIONES

Se recomienda utilizar una dirección ip estática para el broker, en vez de una proporcionada por dhcp, ya que de esta manera se evita estar modificado las configuraciones de los suscriptores

Se recomienda que el modelado de la planta sea lo más exacto posible, ya que los resultados proporcionados por el controlador LQG dependen en gran medida de este proceso.

Es recomendable fijar la cámara a la misma altura del tanque, de tal manera que se pueda obtener un enfoque completo del líquido mejorando así el sensado

Al realizar el sensado de nivel mediante la cámara web, es recomendable el uso de luz artificial ya que la luz natural al variar su intensidad con el paso del día puede afectar la precisión con la que la cámara web obtiene el dato, y esto hace que se tenga que realizar una modificación constante en la calibración

Como una manera de optimizar el proceso realizado, se podría utilizar una cámara enfocada específicamente en visión artificial, ya que este tipo de cámaras cuentan con muchas más aplicaciones y mejoras que la cámara web utilizada en este trabajo, de tal manera que el error de medición podría reducirse y el proceso de sensado mejoraría dando mejores resultados al controlador

REFERENCIAS

Athans, M. (1971). The role and use of the stochastic Linear-Quadratic-Gaussian problem in control system design. *IEEE Transaction on Automatic Control*, 529-552.

Baidez, S. (2018). Monitorización remota de plantas mediante Rasbperry y Telegram. *(Tesis de grado).* Universidad Politécnica de Valencia, Valencia.

BANERJE, R. (2015). Water Level Controller by fuzzy logic. *IJIRAE*.

Bharathi, M., & Kumar, G. (2013). Un enfoque de diseño del controlador LQR para la estabilización del eje de cabeceo del sistema de helicóptero 3-DOF. *Revista Internacional de Investigación Científica e Ingeniería*.

Càceres, J. P. (2011). Sistema de Visiòn Artificial para inspeciòn de nivel de llenado de bebidas embotelladas. *(Tesis de Grado).* Universidad Tècnica de Ambato, Ambato.

CHEN, J., McKERNAN, A., IRWIN, G. W., & SCALON, W. G. (2008). Experimental characterisation and analysis of wireless network control systems. *IET Irish Signals and Systems Conference*, 238–243.

COGNEX.COM. (2020). *COGNEX*. Obtenido de VISION ARTIFICIAL: https://www.cognex.com/es-ar/what-is/machine-vision/what-is-machine-vision

Consulting Informàtico. (07 de Febrero de 2019). *¿Qué es y cómo funciona la monitorización en los procesos industriales?* Obtenido de https://www.cic.es/que-es-la-monitorizacion-en-los-procesos-industriales/#:~:text=La%20monitorizaci%C3%B3n%20no%20es%20m%C3%A1s,Internet%20Industrial%20De%20las%20Cosas%E2%80%9D.

Cumplir los requisitos mínimos de hardware. (21 de 05 de 2020). Obtenido de www.debian.org: https://www.debian.org/releases/jessie/i386/ch03s04.html.es

debian.org. (21 de 05 de 2020). Obtenido de debian.org: https://www.debian.org/releases/jessie/i386/ch03s04.html.es

Deloitte. (2020). *IoT - Internet Of Things*. Obtenido de Dloitte España: https://www2.deloitte.com/es/es/pages/technology/articles/IoT-internet-of-things.html

Eclipse Mosquitto ™. (2020). *Un corredor de código abierto MQTT*. Obtenido de https://mosquitto.org/

EL ECONOMISTA. (21 de 05 de 2020). Obtenido de ¿Cómo funciona Telegram?: https://www.eleconomista.net/tendencias/-Como-funciona-Telegram-y-en-que-se-diferencia-de-WhatsApp-20190705-0019.html

Gonzalo, C. (2016). COMPARACIÓN DE UN CONTROLADOR LQR VS UN CONTROLADOR PID. *(Tesis de ingeniería).* UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS, Bogota.

hivemq.com. (21 de 05 de 2020). *MQTT essentials introducing MQTT*. Obtenido de https://www.hivemq.com/blog/mqtt-essentials-part-1-introducing-mqtt/

INFAIMON. (2017). *Sistemas de visión artificial: tipos y aplicaciones*. Obtenido de INFAIMON: https://blog.infaimon.com/sistemas-de-vision-artificial-tipos-aplicaciones/

Jarquin, R., & Zepeda, S. (2017). Development of a fuzzy controller for liquid level by using Raspberry pi and Internet of Things. *IEEE Central America and Panama Student Conference (CONESCAPAN).*

Justo de Frìas, C. (2019). Visión artificial aplicada en la identificación de objetos y su parametrización geométrica. *(Tesis de fin de grado).* Universidad Carlos III de Madrid, Madrid.

MathWorks. (2020). *Filtros de Kalman*. Obtenido de MathWorks: https://la.mathworks.com/discovery/kalman-filter.html

MATIAKIS, T., HIRCHE, S., & BUSS, M. (2009). Control of networked systems using the scattering transformation. *IEEE Transactions on Control Systems Technology, vol. 17,* 60 – 67.

Menendez, A., & Simon, S. (13 de 12 de 2004). *Aportacion al control del convertidor CC/CA de tres niveles.* Obtenido de Tesis Doctorales en Xarxa:

https://www.tdx.cat/handle/10803/6330;jsessionid=B5BD7A2FC9D032E98CB9CB8B3266D056#page=1

OMEGA ENGINEERING. (Diciembre de 2015). *http://www.reporteroindustrial.com/*. Obtenido de Cómo aprovechar la tecnología para monitoreo inalámbrico en la industria: http://www.reporteroindustrial.com/temas/Como-aprovechar-la-tecnologia-para-monitoreo-inalambrico-en-la-industria+109270?pagina=2

Programo Ergo Sum. (20 de 05 de 2020). Obtenido de Programo Ergo Sum: https://www.programoergosum.com/cursos-online/raspberry-pi/244-iniciacion-a-python-en-raspberry-pi/que-es-python

proximassystems.net. (2018). Obtenido de Industria 4.0: https://www.proximasystems.net/scada-industrias-monitorizacion-industrial/

Ricardo Vega. (07 de 04 de 2015). *NODE-RED: CONSTRUYE EL INTERNET DE LAS COSAS*. Obtenido de ricveal.com: https://ricveal.com/blog/node-red-construye-el-internet-de-las-cosas/

unpocodejava.com. (20 de 05 de 2020). Obtenido de ¿Qué es OpenCV?: https://unpocodejava.com/2013/10/09/que-es-opencv/

www.techedgegroup.com. (20 de 04 de 2020). Obtenido de FUNDAMENTOS DE NODE-RED: https://www.techedgegroup.com/es/blog/fundamenos-node-red

xataka.com. (21 de 05 de 2020). Obtenido de Cómo funciona Arduino: https://www.xataka.com/basics/que-arduino-como-funciona-que-puedes-hacer-uno

ANEXOS

ANEXO A: Programación en Arduino para Adquisición de datos experimentales

Figura A.1 Script “adquisición_de_datos_experimentales”

Programa para adquisición de nivel en Arduino, (Sánchez Bryan & Tupiza Alex)

La descripción del script “adquisición_de_datos_experimentales” se puede apreciar en la Tabla A.1

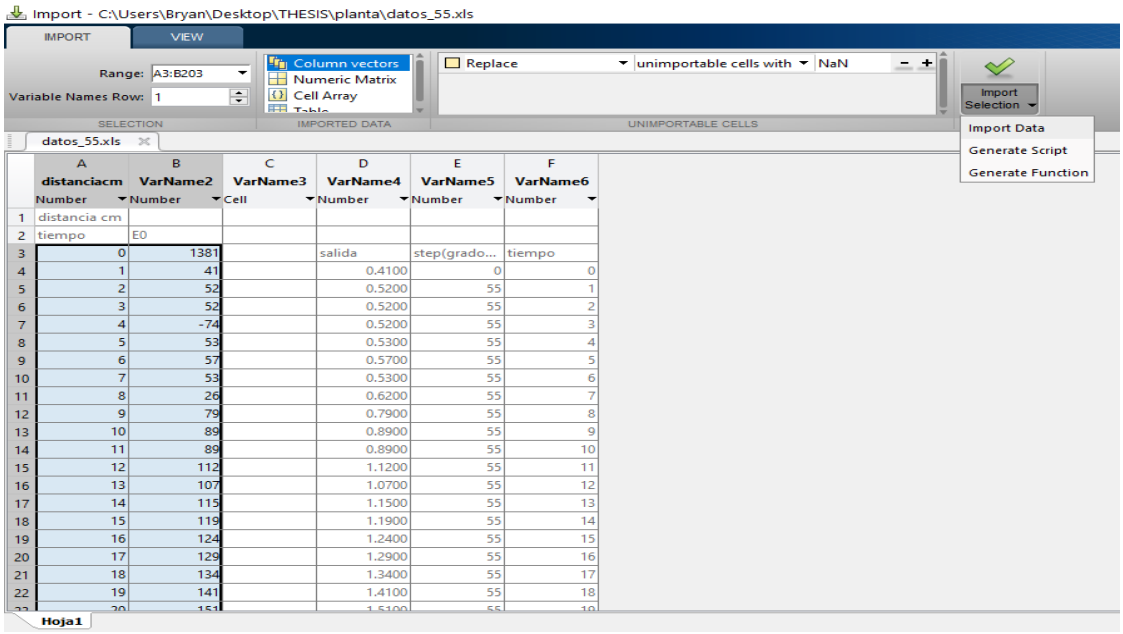
Tabla A.1. Descripción de las funciones del script
"adquisición_de_datos_experimentales"

N	Función	Acción
1	Declaración de variables	- int trigger =10: La variable trigger es de tipo entero y designa al pin digital PWM número 10 como el disparador de un pulso funcionando como onda original en busca del objeto más cercano, que en es el nivel de combustible dentro del tanque de llenado - int echo = 8: La variable echo de tipo entero y designa el pin digital número 8 como el receptor de la onda reflejada por la superficie actual de combustible en el tanque de llenado - String envía="": La variable "envía" es de tipo String y guarda el nivel adquirido por el sensor HC-SR04, para exportar los datos
2	setup	- pinMode(triger,OUTPUT): Declara al pin triger como salida - pinMode(echo,INPUT): Declara a l pin echo como entada - Serial.begin(9600): Este comando inicia comunicación serial con la PC, con una velocidad de 9600 bits por segundo (baudios), los datos transferidos pueden distinguirse en el monitor serial del Arduino IDE
3	Obtención de dato nivel	- digitalWrite(triger,LOW): se encarga de iniciar el pin digital número 10 (triger) en low digitalWrite(triger,HIGH): cambia el estado a high de esta manera se genera el pulso de la onda original que tiene contacto con la superficie más cercana - float tiempo=pulseIn (echo, HIGH): La variable tiempo guarda la cantidad de tiempo en microsegundos que demora el pin digital número 8 (echo) en adquirir la onda reflejada por la superficie del nivel de combustible en el tanque. - float distancia = 19.5-((tiempo/2)/29.1): La variable distancia guarda el valor calculado por la Ec(3.2), esta ecuación describe la distancia desde la base del tanque de llenado hasta el nivel de combustible, se tomó en cuenta la altura en centímetros del tanque de llenado ya que el sensor físicamente está dispuesto en la parte superior del mismo ,como se muestra a continuación $Calculo\ de\ nivel\ en\ tanque\ de\ llenado = 19.5 - \frac{\frac{tiempo\ us}{2}}{29.1\ us}$
4	Registro de datos de nivel por puerto serial	- envia.concat (distancia): Utilizando el método naive con la función .concat se realizó la concatenación para adquirir el dato guardado en la variable tipo string "envia", además se borra el valor en la siguiente línea para calcular la nueva distancia en la siguiente lectura y no se sobrescriba el valor. - Serial.println (envia): La función Serial.println() envía el valor a el monitor serial de arduino para verificar que el sensor HC-SR04 funcione de manera correcta.

Tabla de descripción de las funciones del script para adquirir los datos experimentales de nivel,
(Sánchez Bryan & Tupiza Alex)

ANEXO B: Obtención de la función de transferencia, mediante el software Matlab

Figura B.1 Importación a Matlab en forma de variables



Interfaz de Matlab, (Sánchez Bryan & Tupiza Alex)

Figura B.2 Parámetros de la ventana “Process Model”

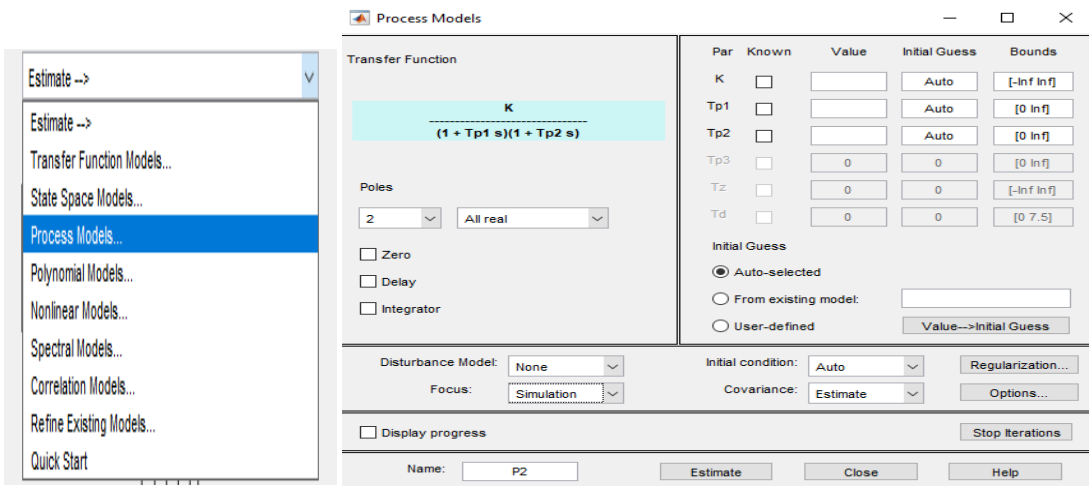
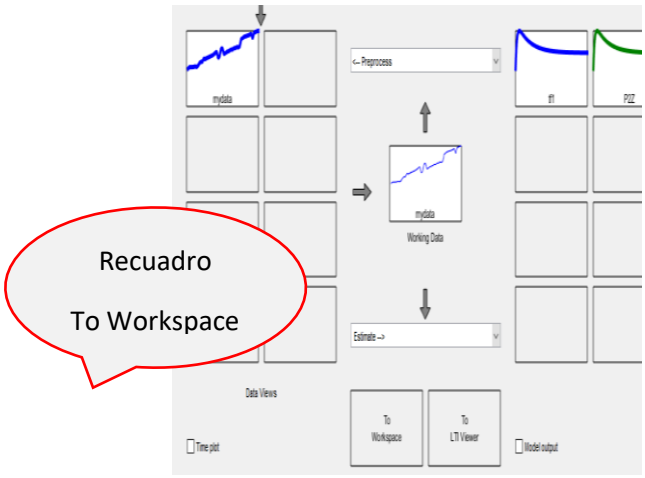


Figura B.3 Parámetros de la ventana “Process Model”




```
tf1 =

  From input "u1" to output "y1":
  -0.003263 s + 0.003073
  ----------------------
  s^2 + 0.583 s + 0.02797

Name: tf1
Continuous-time identified transfer function.

Parameterization:
   Number of poles: 2   Number of zeros: 1
   Number of free coefficients: 4
   Use "tfdata", "getpvec", "getcov" for parameters and their uncertainties.

Status:
Estimated using TFEST on time domain data "mydata".
Fit to estimation data: 89.81% (simulation focus)
FPE: 0.02531, MSE: 0.02387
```

Interfaz de IDENT de Matlab, (Sánchez Bryan & Tupiza Alex)

ANEXO C: Algoritmo diseñado para el script de vision artificial

Figura C.1 Algoritmo de visión artificial

```
import cv2 as cv
import numpy as np

cap = cv.VideoCapture(0)
def nothing(x):
    pass
cv.namedWindow("Calibracion",cv.WINDOW_NORMAL)
cv.createTrackbar("l-h","Calibracion",0,179,nothing)
cv.createTrackbar("l-s","Calibracion",0,255,nothing)
cv.createTrackbar("l-v","Calibracion",0,255,nothing)
cv.createTrackbar("L-H","Calibracion",179,179,nothing)
cv.createTrackbar("L-S","Calibracion",255,255,nothing)
cv.createTrackbar("L-V","Calibracion",255,255,nothing)
b=440
maxi1 = b
r=100
nivel_porcentaje1 = 0
font = cv.FONT_HERSHEY_SIMPLEX
```

```
kernel = np.ones((5,5),np.uint8)
while w == 1:
      _,frame = cap.read()
      l_h=cv.getTrackbarPos("l-h","Calibracion")
      l_s=cv.getTrackbarPos("l-s","Calibracion")
      l_v=cv.getTrackbarPos("l-v","Calibracion")
      L_H=cv.getTrackbarPos("L-H","Calibracion")
      L_S=cv.getTrackbarPos("L-S","Calibracion")
      L_V=cv.getTrackbarPos("L-V","Calibracion")
      frame1=frame[0:480,300:530]
      base = int (0)
      base_down = int(b)
      blurred_frame1=cv.GaussianBlur(frame1,(15,15),0)
```

```
    hsv_rojo1= cv.cvtColor(blurred_frame1, cv.COLOR_BGR2HSV)
    lower_rojo1=np.array([l_h,l_s,l_v])
    upper_rojo1= np.array([L_H,L_S,L_V])
    mask1= cv.inRange(hsv_rojo1,lower_rojo1,upper_rojo1)
    r1=cv.bitwise_and(frame1,frame1,mask=mask1)
    cv.imshow("bits1",r1)
    _,contours1,_= cv.findContours(mask1, cv.RETR_TREE, cv.CHAIN_APPROX_SIMPLE)
          for x1 in range(len(contours1)):
              area1= cv.contourArea(contours1[x1])
              if area1>r:
                  maxi1 = contours1[x1][:,:,1].argmin()
                  maxi1 = contours1[x1][maxi1][0][1]
                  nivel_porcentaje1 = (base_down-maxi1)*100/(base_down-base)
                  nivel_porcentaje1 = nivel_porcentaje1*0.2
                  nivel_porcentaje1 = round(nivel_porcentaje1,2)
                  print  (nivel_porcentaje1)
```

```
          cv.putText(frame,str(nivel_porcentaje1),(410,maxi1-7), font, 1,(255,0,0),2)
          cv.line(frame,(290,maxi1),(490,maxi1),(0,255,0),3)
          cv.imshow("Niveles",frame)
          k = cv.waitKey(1) & 0xFF
          if k == 32 :
                break
    cv.destroyAllWindows()
bot.polling(none_stop = False)
```

Algoritmo en python de visión artificial, (Sánchez Bryan & Tupiza Alex)

Tabla C.1 Descripción de las funciones del script “visión.py”

N°	Función	Acción
1	Importación de librerías	- import cv2 as cv: OpenCV se importada al script con el comando import y sirve para reconocer el escenario que es sensado, es decir el tanque de llenado y monitorea el nivel del mismo. - import numpy as np: Numpy es la librería que dimensiona datos estructurados, se usa para trabajar las imágenes que se procesan y están dimensionadas en forma de matriz.
2	Inicio de cámara	- cap = cv.VideoCapture (0): La variable “cap” inicia la captura de imágenes con el método cv.VideoCapture (0) en el numero puerto USB de la Raspberry que se asigna entre paréntesis.
3	Creación de la ventada de calibración	- def nothing(x): La función nothing crea la ventada de calibración, esta función solo se dedica a capturar la imagen es decir no retorna ningún valor. - cv.namedWindow("Calibracion",cv.WINDOW_NORMAL) cv.createTrackbar("l-h","Calibracion",0,179,nothing) cv.createTrackbar("l-s","Calibracion",0,255,nothing)

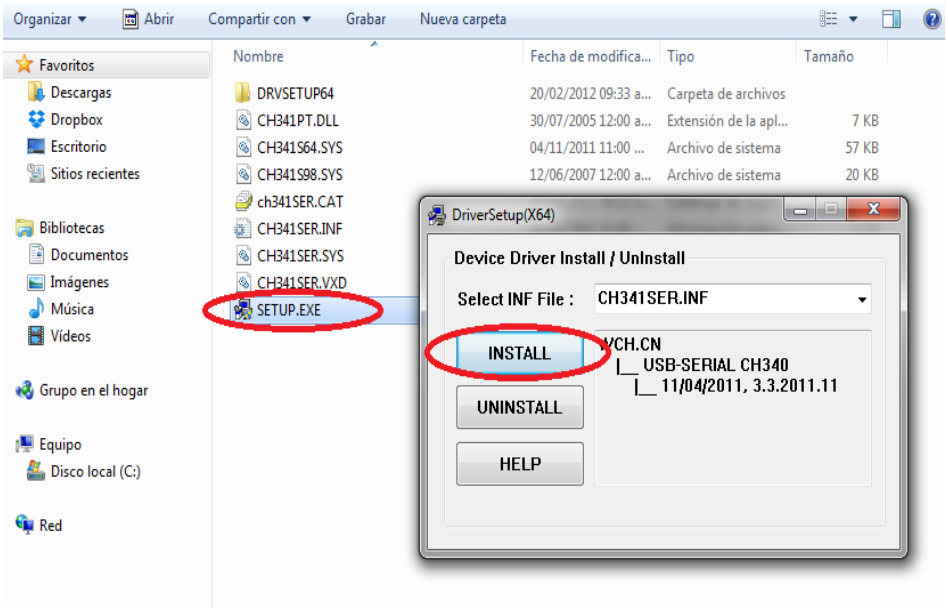
		cv.createTrackbar("l-v","Calibracion",0,255,nothing) cv.createTrackbar("L-H","Calibracion",179,179,nothing) cv.createTrackbar("L-S","Calibracion",255,255,nothing) cv.createTrackbar("L-V","Calibracion",255,255,nothing) Las funciones cv.create crean sliders en la ventana "Calibracion" con sus valores mínimos y máximos para configurar el tono de color que se quiere captar para monitorear el nivel, las imágenes son capturadas en el rango OpenCV HSV que es: H: 0 a 179 S: 0 a 255 V: 0 a 255
4	Declaración de dimensiones del área de imagen a procesar	- b=440: La variable b asigna el número de pixeles, definiendo el límite inferior del frame de la imagen que se trabaja. - maxi1 = b: La variable maxi1 asigna la posición máxima en la imagen que se trabaja. - r=100: La variable r asigna el reflejo necesario para medir el nivel, es decir los pixeles que no reflejan el color detectado. - nivel_porcentaje1 = 0: La variable nivel_porcentaje1 almacena el nivel que se detecta después de haber realizado el algoritmo de visión artificial. - font = cv.FONT_HERSHEY_SIMPLEX: La variable font determina la fuente de la letra que muestra el nivel del tanque de llenado en la pantalla de resultado "niveles". - kernel = np.ones((5,5),np.uint8): La variable kernel elimina el ruido de la imagen en la que se trabaja.
5	Ejecución de ciclo while	_, frame = cap.read (): El método _, frame empieza la captura del video constante. l_h=cv.getTrackbarPos("l-h","Calibracion") l_s=cv.getTrackbarPos("l-s","Calibracion") l_v=cv.getTrackbarPos("l-v","Calibracion") L_H=cv.getTrackbarPos("L-H","Calibracion") L_S=cv.getTrackbarPos("L-S","Calibracion") L_V=cv.getTrackbarPos("L-V","Calibracion") Las funciones cv.get llaman a los datos conseguidos en los sliders de calibración de color y asignan estos datos a las variables l_h, l_s, l_v, L_H, L_S, L_V. - frame1=frame[0:480,300:530]: La variable frame1 crea el recuadro para analizar únicamente el área que le corresponde al tanque de llenado, es decir limita la imagen al área de interés. - base = int (0) , base_down = int(b) Las variables base y base_down guarda los limites superior e inferior en pixeles respectivamente. - blurred_frame1=cv.GaussianBlur (frame1,(15,15),0): blurred_frame aplica el filtro gaussiano a la variable frame1.

		- hsv_rojo1=cv.cvtColor(blurred_frame1,cv.COLOR_BGR2HSV):hsv_rojo1 cambia de formato de RGB a formato HSB Las variables lower_rojo 1 y upper_rojo1 rellena la matriz numpy, filas y columnas respectivamente del color configurado en los sliders. - La variable mask1 crea una marcara binaria en negro y blanco del frame1 y r1 sobrepone la máscara en el contorno del tanque de llenado. - cv.imshow("bits1",r1): cv.imshow muestra el contorno del color detectado en la captura de imagen - _,contours1,_=cv.findContours(mask1,cv.RETR_TREE,cv.CHAIN_APPROX_SIMPLE: Esta variable busca y guarda la cantidad de pixeles que conforman el contorno en el frame 1.
6	Recorrido de contorno en el frame1	- for x1 in range(len(contours1)): area1= cv.contourArea(contours1[x1]) La sentencia for recorre la cantidad de pixeles en los contornos que se han encontrado y la variable área1 guarda todos los datos - if area1>r: maxi1 = contours1[x1][:,:,1].argmin() maxi1 = contours1[x1][maxi1][0][1] La sentencia if compara el color de los pixeles encontrados con el color configurado en los sliders y la variable maxi1 encuentra la coordenada máxima del contorno, extrayendo el dato en el eje Y. - nivel_porcentaje1 = (base_down-maxi1)*100/(base_down-base) nivel_porcentaje1 = nivel_porcentaje1*0.2 - nivel_porcentaje1 = round(nivel_porcentaje1,2) La variable nivel_porcentaje1 calcula el nivel en el tanque de llenado en base a una regla de tres tomando en cuenta la coordenada en Y, además redondea el nivel con 2 cifras para que el monitoreo sea especifico.
7	Visualización de nivel	- cv.putText(frame,str(nivel_porcentaje1),(410,maxi1-7), cv.line(frame,(290,maxi1),(490,maxi1),(0,255,0),3) cv.imshow("Niveles",frame) El nivel se muestra interactivamente en la pantalla “Niveles” en forma de texto con la fuente seleccionada en tiempo real, el nivel además se muestra con una línea horizontal verde debajo del texto. - break :El comando break interrumpe el ciclo infinito del programa dando paso el comando c.destroy que cierra todas las ventanas que se encuentren abiertas

Tabla de descripción de las funciones del script de visión artificial en python, (Sánchez Bryan & Tupiza Alex)

ANEXO D: Configuracion y diseño del algoritmo para el modulo ESP8266

Figura D.1 Instalación del driver CH341SE



Interfaz de driver CH341SE, (Sánchez Bryan & Tupiza Alex)

Figura D.2. Algoritmo del módulo ESP8266 Node.

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <OneWire.h>
#include <Servo.h>
Servo myservo;
const char* ssid = "NETLIFE-DALILA";
const char* password = "1708948904";
const char* mqtt_server = "192.168.100.124";
WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[50];
int value = 0;
void setup() {
  Serial.begin(9600);
  myservo.attach(2);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}
void setup_wifi() {
  delay(10);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}
```

```
06 void callback(char* topic, byte* payload, unsigned int length) {
     String string;
   Serial.print("Message arrived [");
   Serial.print(topic);
   Serial.print("] ");
   for (int i = 0; i < length; i++) {
    // Serial.print((char)payload[i]);
       string+=((char)payload[i]);
   }
```

```
07 void reconnect() {
   // Loop until we're reconnected
   while (!client.connected()) {
     Serial.print("Attempting MQTT connection...");
     // Attempt to connect
     if (client.connect("ESP8266Client")) {

       Serial.println("connected");
       client.subscribe("servo");
     } else {
       Serial.print("failed, rc=");
       Serial.print(client.state());
       Serial.println(" try again in 5 seconds");
       // Wait 5 seconds before retrying
       delay(5000);
     }}}
```

```
08 void loop() {
     if (!client.connected()) {
     reconnect(); }
     client.loop();
    delay(100);
   }
```

Algoritmo de conexión Wi-fi, (Sánchez Bryan & Tupiza Alex)

La descripción del script “conectar_servo.ino” se pueden apreciar en la siguiente tabla

Tabla D.1. Descripción de las funciones del script “conectar_servo.py”

N°	Función	Acción
1	Importación de librerías	- #include <ESP8266WiFi.h>: La librería ESP8266WiFi.h es utilizada para habilitar la antena Wi-Fi del módulo y comenzar la comunicación inalámbrica con el sistema. - #include <PubSubClient.h>: La librería PubSubClient.h habilita el dispositivo como un cliente más en el protocolo MQTT y le da la posibilidad de publicar o suscribirse al topic de su interés. - #include <Servo.h>: La librería Servo.h escribe directamente los grados de apertura o cierre determinados por el control LQG en el servo HS3-11.
2	Declaración de constantes	- Servo myservo: Guarda los grados de apertura que el actuador va a ejercer por el algoritmo de control LQG - const char* ssid: El dato ssid es una variable tipo char que guarda el nombre de la red LAN que aloja sistema - const char* password: El dato password es una variable tipo char que guarda la contraseña que la red local tiene para restringir la conexión de los dispositivos.

		- const char* mqtt_server: El dato mqtt_server guarda la dirección donde se ejecuta el bróker MQTT en la tarjeta Raspberry. - WiFiClient espClient: El parámetro espClient define al módulo como un cliente en el protocolo MQTT. - PubSubClient client(espClient): El parámetro client() asigna la función de subscribir y publicar al nuevo cliente que en este caso es el módulo ESP8266.
3	Función setup	- Serial.begin(9600); - setup_wifi() - client.setServer(mqtt_server, 1883); La función setup define el orden en el que las funciones van a ejecutarse en el programa y también define los siguientes parámetros respecto a la conectividad: Velocidad de conexión de 9600 baudios, número de pin (2) de salida, puerto lógico por el que el servidor escucha al cliente, en este caso es el 1883.
4	Función setup_wifi	- Serial.println(): Los comandos serial.print muestran la red a la que el modulo se conecta en el monitor serial de arduino - WiFi.begin(ssid, password): El método Wifi.begin inicializa la búsqueda de la red local
5	Verificación de la conexión del modulo	- while (WiFi.status() != WL_CONNECTED) { El ciclo while muestra el estado de conectividad del módulo en la red local. Los comandos serial.print confirman la conexión del dispositivo, mostrando el nombre de la red y la IP asignada.
6	Obtención de dato de accionamiento para el servo	- void callback(char* topic, byte* payload, unsigned int length){ EL método callback recibe el dato de accionamiento para el servo desde el cliente que lo publica. El callback toma en cuenta el topic al cual se está suscrito y también el nombre del mensaje que quiere recibir en este caso su nombre es payload. - La sentencia for recorre el dato payload y guarda en una variable tipo string el dato requerido. - La sentencia if compara el topic para que el cliente haga un mapeo del dato de accionamiento para transmitir la apertura o cierre del mismo.

7	Función de reconexión del módulo	- void reconnect() {
		while (!client.connected()) {
		Serial.print("Attempting MQTT connection...");
		La función reconnect se habilita en caso de desconexión del módulo a la red local Wi-Fi, asegura el intercambio de datos con Node Red.
		La sentencia if reconoce como cliente MQTT al módulo y muestra el estado de conexión y el dato que recibe el servo en el monitor serial de arduino.

8	Ciclo de repetición	- void loop() {
		if (!client.connected()) { reconnect(); }
		client.loop(); delay(100); }
		Void loop ordena las funciones para que el cliente se reconecte en caso de fallo de conexión repitiendo la verificación del mismo cada 100 milisegundos.

Tabla de descripción de las funciones del script de accionamiento inalámbrico del módulo ESP8266, (Sánchez Bryan & Tupiza Alex)

ANEXO E: DATOS OBTENIDOS DE LA PLANTA

Tabla E.1. Datos de nivel en 3.5 cm para determinar la exactitud de medida

N°	nivel [cm]	Error absoluto [cm]	Error relativo (%)
1	3,45	0,050	1,429
2	3,5	0,000	0,000
3	3,5	0,000	0,000
4	3,59	-0,090	-2,571
5	3,5	0,000	0,000
6	3,59	-0,090	-2,571
7	3,59	-0,090	-2,571
8	3,59	-0,090	-2,571
9	3,55	-0,050	-1,429
10	3,5	0,000	0,000
11	3,59	-0,090	-2,571
12	3,64	-0,140	-4,000
13	3,45	0,050	1,429
14	3,59	-0,090	-2,571
15	3,36	0,140	4,000
16	3,5	0,000	0,000
17	3,5	0,000	0,000
18	3,45	0,050	1,429
19	3,55	-0,050	-1,429
20	3,59	-0,090	-2,571
suma	70,58	-0,58	-16,571
promedio	3,529	-0,029	-0,829

Tabla de datos obtenida para realizar pruebas de exactitud a una medida de 3.5cm con la ayuda del script de visión artificial y Excel (Sánchez Bryan & Tupiza Alex)

Tabla E.2. Resultados del tiempo de subida, estabilización y máximo sobre impulso en los distintos niveles para el Control PID

CONTROL PID				
N° DATO	NIVEL	TIEMPO DE SUBIDA	TIEMPO DE ESTABILIZACION	MÁXIMO SOBRE IMPULSO
1	0 A 4	10	30,75	0,23
2	4 A 8	22	45	0,27
3	8 A 12	23	42	0,75
4	12 A 16	17,25	43,25	0,27
5	16 A 18	28,5	43,75	0,14
6	18 A 16	5,25	15,95	0,41
7	16 A 12	12,5	23,25	0,5
8	12 A 8	11,5	20,25	0,75
9	8 A 4	13,5	30,75	1
10	4 A 8	24,75	39,5	0,5
11	8 A 12	32,25	39,75	0,27
12	12 A 16	20,25	40,75	0,23
13	16 A 18	20,5	26,75	0,25
14	18 A 16	2,25	15,91	0,45
15	16 A 12	7,75	16,75	0,18
16	12 A 8	9	19,5	0,41
17	8 A 4	8,5	20,5	0,45
18	4 A 8	26,25	40,25	0,23
19	8 A 12	20	30,25	0,15
20	12 A 16	33,5	43,25	0,18
21	16 A 18	18	27,25	0,5
22	18 A 16	10	26,75	0,41
23	16 A 12	6,75	16,25	0,55
24	12 A 8	12,25	21,25	0,45
25	8 A 4	15	27,25	0,27

Tabla de pruebas de control PID en diferentes niveles, (Sánchez Bryan & Tupiza Alex)

Tabla E.3. Resultados del tiempo de subida, estabilización y máximo sobre impulso en los distintos niveles para el Control LQG

CONTROL LQG				
N° DATO	NIVEL [cm]	TIEMPO DE SUBIDA [s]	TIEMPO DE ESTABILIZACION	MÁXIMO SOBRE IMPULSO [cm]
1	0 A 4	8,25	10	0,14
2	4 A 8	11,25	13,5	0,1
3	8 A 12	13	15	0,07
4	12 A 16	16,5	20	0,08
5	16 A 18	10	13,75	0,1
6	18 a 16	4,5	14	0,38
7	16 A 12	2,5	5,25	0,06
8	12 A 8	5,25	7,5	0,21
9	8 A 4	7,75	10,25	0,1
10	4 A 8	11,75	13,5	0,1
11	8 A 12	14	16,75	0,07
12	12 A 16	16	20	0,12
13	16 A 18	7,5	11	0,1
14	18 a 16	2	7,25	0,33
15	16 A 12	3,25	5,75	0,29
16	12 A 8	5,25	7,75	0,25
17	8 A 4	8	12	0,13
18	4 A 8	11,75	14	0,06
19	8 A 12	13,5	15,5	0,071
20	12 A 16	15,5	20,5	0,08
21	16 A 18	7,5	14,75	0,11
22	18 a 16	1,75	11	0,33
23	16 A 12	2,75	6	0,25
24	12 A 8	5,5	8,5	0,25
25	8 A 4	9	11,25	0,13

Tabla de pruebas de control LQG en diferentes niveles, (Sánchez Bryan & Tupiza Alex)

Tabla E.4. Prueba de medición con 30 datos en los niveles de 4cm, 8cm, 12cm, 16cm y 18cm para el controlador PID

CONTROL PID					
MEDIDA [cm]	4	8	12	16	18
DATO	VALORES MEDIDOS [cm]				
1	4,05	8,05	12	16,05	18,09
2	4,14	8,09	12,05	16,05	18,14
3	4,23	8,09	12,09	16,09	17,91
4	4,09	8,27	12,09	16,14	18,14
5	4,14	8,27	12,14	16,18	18,09
6	4,14	8,27	12,18	16,18	18
7	3,91	8,23	12,18	16,14	17,77
8	4,05	8,18	12,05	16,05	17,77
9	3,95	8,18	12	16,05	17,64
10	4	8,14	12	15,82	17,59
11	3,95	8	11,91	15,59	17,68
12	4,05	8	11,86	15,59	17,73
13	3,86	7,95	11,73	15,41	17,77
14	3,77	7,91	11,68	15,36	17,91
15	3,73	7,91	11,59	15,5	17,73
16	3,59	7,86	11,55	15,59	17,91
17	3,68	7,77	11,55	15,59	17,77
18	3,64	7,73	11,68	15,59	17,95
19	3,59	7,59	11,77	15,68	17,91
20	3,73	7,64	11,86	15,68	18
21	3,82	7,64	11,95	15,73	17,77
22	3,68	7,73	12,09	15,73	17,91
23	3,77	7,86	12,09	15,82	17,82
24	3,73	7,91	11,91	15,82	17,86
25	3,86	7,95	12,14	15,82	17,86
26	3,64	8	12,18	15,86	17,73
27	3,91	8	12,18	15,86	18
28	3,77	8,05	12,23	15,95	17,91
29	3,86	8,05	11,86	15,86	17,91
30	3,82	8,05	11,86	16	18

Tabla de datos obtenida para realizar pruebas con el controlador PID (Sánchez Bryan & Tupiza Alex)

Tabla E.5 Prueba de medición con 30 datos en los niveles de 4cm, 8cm, 12cm, 16cm y 18cm para el controlador LQG

CONTROL LQG					
MEDIDA [cm]	4	8	12	16	18
DATO	VALORES MEDIDOS [cm]				
1	4,0542	8,01824	12,033	16,03682	18,1084
2	4,1434	8,06953	12,122	16,08588	18,1081
3	4,1036	7,84006	11,805	15,85759	17,7908
4	3,9152	7,86168	11,8437	16,12431	17,931
5	3,8866	7,90171	11,893	16,12712	18,0191
6	3,8252	8,0909	12,1214	15,90794	17,9288
7	3,9154	8,01824	12,2105	15,71955	17,7018
8	4,0541	8,06953	12,2099	15,76961	17,7024
9	4,0548	8,06692	12,2997	15,80976	17,7509
10	3,9646	8,0678	12,033	15,85807	17,7908
11	3,8253	7,88041	12,122	15,90738	17,8416
12	3,9646	8,01824	11,805	15,99589	17,8906
13	4,0141	8,06953	12,0317	16,03712	17,9793
14	4,1428	7,83996	12,0723	16,03679	18,0192
15	3,9645	7,92822	11,8437	15,88016	18,0692
16	3,7864	8,01779	11,8775	15,65257	17,9292
17	3,8752	8,01788	11,926	15,71274	18,0196
18	4,0148	7,83894	11,9502	15,85239	17,9292
19	4,0135	8,01908	11,7539	15,94129	17,8906
20	3,915	8,10792	11,8442	16,19105	17,8398
21	3,738	7,89002	11,8437	15,82943	17,9793
22	3,7868	7,83937	11,9819	15,6798	17,9796
23	3,9169	7,93009	12,0316	15,87931	18,0691
24	4,0135	8,01908	12,0326	15,97101	18,1079
25	4,0549	8,10792	11,8435	16,05782	17,8895
26	3,8754	7,83993	12,0334	16,21834	17,9302
27	3,7362	7,90079	12,0722	15,94731	18,0196
28	3,7858	7,95083	11,9665	15,99774	17,79
29	3,8757	8,01908	11,8056	16,03732	17,5726
30	3,9655	8,10792	11,9436	16,08577	17,524

Tabla de datos para realizar pruebas con el controlador LQG (Sánchez Bryan & Tupiza Alex)

Tabla E.6. Tabla de análisis para prueba de Wilcoxon en el tiempo de subida

N° DATO	NIVEL [cm]	T. DE SUBIDA EN CONDICION 1 (PID) [s]	T. DE SUBIDA EN CONDICION 2 (LQG) [s]	DIFERENCIA DE C1 -C2	DIFERENCIA ABSOLUTA	RANKING
1	0 A 4	10	8,25	1,75	1,75	1
2	4 A 8	22	11,25	10,75	10,75	2
3	8 A 12	23	13	10	10	3
4	12 A 16	17,25	16,5	0,75	0,75	4
5	16 A 18	28,5	10	18,5	18,5	5
6	18 a 16	5,25	4,5	0,75	0,75	6
7	16 A 12	12,5	2,5	10	10	7
8	12 A 8	11,5	5,25	6,25	6,25	8
9	8 A 4	13,5	7,75	5,75	5,75	9
10	4 A 8	24,75	11,75	13	13	10
11	8 A 12	32,25	14	18,25	18,25	11
12	12 A 16	20,25	16	4,25	4,25	12
13	16 A 18	20,5	7,5	3,75	13	13
14	18 a 16	2,25	2	0,5	0,25	14
15	16 A 12	7,75	3,25	4,5	4,5	15
16	12 A 8	9	5,25	3,75	3,75	16
17	8 A 4	8,5	8	0,5	0,5	17
18	4 A 8	26,25	11,75	14,5	14,5	18
19	8 A 12	20	13,5	6,5	6,5	19
20	12 A 16	33,5	15,5	18	18	20
21	16 A 18	18	7,5	10,5	10,5	21
22	18 a 16	10	1,75	8,25	8,25	22
23	16 A 12	6,75	2,75	4	4	23
24	12 A 8	12,25	5,5	6,75	6,75	24
25	8 A 4	15	9	6	6	25

Tabla de operaciones realizadas con la prueba de Wilcoxon para el tiempo de subida, (Sánchez Bryan & Tupiza Alex)

Tabla E.7 Tabla de análisis para prueba de Wilcoxon en el tiempo de estabilización

N° DATO	NIVEL [cm]	TIEMPO DE ESTABILIZACION EN CONDICION 1 (PID) [s]	TIEMPO DE ESTABILIZACION EN CONDICION 2 (LQG) [s]	DIFERENCIA C1-C2	DIFERENCIA ABSOLUTA	RANKING
1	0 A 4	30,75	10	20,75	20,75	1
2	4 A 8	45	13,5	31,5	31,5	2
3	8 A 12	42	15	27	27	3
4	12 A 16	43,25	20	23,25	23,25	4
5	16 A 18	43.75	13,75	30	30	5
6	18 a 16	15,95	14	1,95	1,95	6
7	16 A 12	23,25	5,25	18	18	7
8	12 A 8	20,25	7,5	12,75	12,75	8
9	8 A 4	30,75	10,25	20,5	20,5	9
10	4 A 8	39,5	13,5	26	26	10
11	8 A 12	39,75	16,75	23	23	11
12	12 A 16	40,75	20	20,75	20,75	12
13	16 A 18	26,75	11	15,75	15,75	13
14	18 a 16	15,91	7,25	8,66	8,66	14
15	16 A 12	16,75	5,75	11	11	15
16	12 A 8	19,5	7,75	11,75	11,75	16
17	8 A 4	20,5	12	8,5	8,5	17
18	4 A 8	40,25	14	26,25	26,25	18
19	8 A 12	30,25	15,5	14,75	14,75	19
20	12 A 16	43,25	20,5	22,75	22,75	20
21	16 A 18	27,25	14,75	12,5	12,5	21
22	18 a 16	26,75	11	15,75	15,75	22
23	16 A 12	16,25	6	10,25	10,25	23
24	12 A 8	21,25	8,5	12,75	12,75	24
25	8 A 4	27,25	11,25	16	16	25

Tabla de operaciones realizadas con la prueba de Wilcoxon para el tiempo de estabilización, (Sánchez Bryan & Tupiza Alex)

Tabla E.8. Tabla de análisis para prueba de Wilcoxon en máximo sobre impulso

N° DATO	NIVEL [cm]	MAXIMO SOBRE IMPULSO EN CONDICION 1 (PID) [cm]	MAXIMO SOBRE IMPULSO EN CONDICION 2 (LQG) [cm]	DIFERENCIA C1 -C2	DIFERENCIA ABSOLUTA	RANKING
1	0 A 4	0.18	0.29	-0.11	0.11	15
2	4 A 8	0.41	0.38	0.03	0.03	6
3	8 A 12	0.14	0.1	0.04	0.04	5
4	12 A 16	0.15	0.71	0.079	0.079	19
5	16 A 18	0.41	0.33	0.08	0.08	22
6	18 a 16	0.23	0.14	0.09	0.09	1
7	16 A 12	0.18	0.08	0.1	0.1	20
8	12 A 8	0.23	0.12	0.11	0.11	12
9	8 A 4	0.45	0.33	0.12	0.12	14
10	4 A 8	0.27	0.13	0.14	0.14	25
11	8 A 12	0.25	0.1	0.15	0.15	13
12	12 A 16	0.41	0.25	0.16	0.16	16
13	16 A 18	0.27	0.1	0.17	0.17	2
14	18 a 16	0.23	0.06	0.17	0.17	18
15	16 A 12	0.27	0.08	0.19	0.19	4
16	12 A 8	0.27	0.07	0.2	0.2	11
17	8 A 4	0.45	0.25	0.2	0.2	24
18	4 A 8	0.55	0.25	0.3	0.3	23
19	8 A 12	0.45	0.13	0.32	0.32	17
20	12 A 16	0.5	0.11	0.39	0.39	21
21	16 A 18	0.5	0.1	0.4	0.4	10
22	18 a 16	0.5	0.06	0.44	0.44	7
23	16 A 12	0.75	0.21	0.54	0.54	8
24	12 A 8	0.75	0.07	0.68	0.68	3
25	8 A 4	1	0.1	0.9	0.9	9

Tabla de operaciones realizadas con la prueba de Wilcoxon para el máximo sobre impulso, (Sánchez Bryan & Tupiza Alex)